Plant Disease Detection

SML project Btech 2022-2026

Roshan Kumar Mahto 2022418 Tarandeep Singh 2022536

Problem being addressed:

This project on plant disease detection using a leaf dataset with 38 classes and CNN model training is immensely useful due to its potential to revolutionize agricultural practices. By leveraging advanced machine learning techniques, the project enables early and accurate identification of plant diseases, thereby promoting enhanced crop health monitoring, precision agriculture, and economic sustainability. It not only empowers farmers and agricultural experts with actionable insights for proactive disease management but also contributes to global food security efforts by ensuring sustainable crop production. Additionally, the integration of cutting-edge technologies like Convolutional Neural Networks showcases the ongoing digital transformation in agriculture, highlighting the project's role in fostering innovation and resilience within the agricultural sector.

Relevant Literature:

Several key studies in the literature showcase the effectiveness of Convolutional Neural Networks (CNNs) for plant disease detection. For instance,

- **Maruthi et al. (2017)** demonstrated the application of CNNs in accurately identifying plant diseases from leaf images, highlighting their superior performance compared to traditional methods.

- **Kumar et al. (2020)** emphasized the importance of large-scale datasets and preprocessing techniques in improving CNN-based disease identification systems. Additionally, review articles like Kumar et al. (2021) provide a comprehensive overview of deep learning algorithms, including CNNs, used in plant disease detection tasks, discussing their advantages and challenges.

- **Abhay and Mohan (2018) and Akshaya et al. (2020)** also contributed significantly by evaluating different CNN architectures and image processing techniques for robust plant disease classification. These studies collectively underscore the growing prominence of CNNs in revolutionizing automated plant disease diagnosis and management

S. S. Sannakki and V. S. Rajpurohit proposed a method called "Classification of Pomegranate Diseases Based on Back Propagation Neural Network." They focused on segmenting the affected area and used color and texture as features for classification. Their approach involved converting images to L*a*b format to extract chromaticity layers, enhancing classification accuracy to 97.30%. However, a drawback is that this method is limited to specific crops.

Methodology:

Data Preparation : In the data preparation phase using Keras ,a batch size of 32 is set to efficiently process training and validation samples. The images are resized to a standardized 128 x 120 pixel size, maintaining aspect ratio for image quality. Conversion to RGB format ensures consistent color representation across the dataset, crucial for accurate model training.

Data count: Total train data set is 70295 belonging to 38 classes that help the deep learning network to have enough data and model give good accuracy for validation data set as well as test data set and total validation data set is 17572 files belonging to 38 classes.

Model Architecture Design (CNN): .

Construct convolutional layers with appropriate filter sizes and activation functions to extract meaningful features from the input images.Apply max-pooling layers to downsample feature maps and reduce computational complexity.Incorporate dropout layers to prevent overfitting by randomly deactivating neurons during training.

Design a fully connected neural network with an initial layer of 1400 neurons, gradually reducing the number of neurons towards the output layer with 38 neurons corresponding to the 38 disease classes.

Training Configuration:

Set the learning rate to 0.0001, which is considered suitable for stable and gradual model convergence without large fluctuations

Utilize a categorical cross-entropy loss function for multi-class classification, optimizing the model using the Adam optimizer or a similar algorithm.

Divide the dataset into training, validation, and test sets, allocating a significant portion for training to ensure model generalization.

Train the model for 10 epochs, monitoring training and validation metrics such as accuracy, loss, precision, recall, and F1-score.

Experimental Settings:

First, we convert RGB images into digits using Keras with a batch size of 32. If we need faster preprocessing, we would use a batch size of 64, but our system cannot handle that, so we stick to a batch size of 32.

CNN Architecture Design:

1. **Layer 1 (Convolutional):**

    1. Convolutional layer with 2, 32 filters of size 3x3.
    2. Padding set to "same" to preserve spatial dimensions.
    3. Activation function (e.g., ReLU) applied after convolution.
    4. Polling that is max polling and size 2*2 and stride is 2

2. **Layer 2 (Convolutional):**

    1. Convolutional layer with 2, 64 filters of size 3x3.
    2. Padding set to “valid " if keep same then dimension is increase means feature will be increase
    3. Activation function (e.g., ReLU) applied after convolution.
    4. Polling that is max polling and size 2*2 and stride is 2

Layer 3 (Convolutional):

1. Convolutional layer with 2, 128 filters of size 3x3.
2. Padding set to “valid " if keep same then dimension is increase means feature will be increase
3. Activation function (e.g., ReLU) applied after convolution.
4. Polling that is max polling and size 2*2 and stride is 2

Layer 4 (Convolutional):

1. Convolutional layer with 2, 256 filters of size 3x3.
2. Padding set to “valid " if keep same then dimension is increase means feature will be increase
3. Activation function (e.g., ReLU) applied after convolution.

4. Polling that is max polling and size 2*2 and stride is 2

Layer 5 (Convolutional):

1. Convolutional layer with 2, 512 filters of size 3x3.

2. Padding set to “valid " if keep same then dimension is increase means feature will be increase

3. Activation function (e.g., ReLU) applied after convolution.

4. Polling that is max polling and size 2*2 and stride is 2

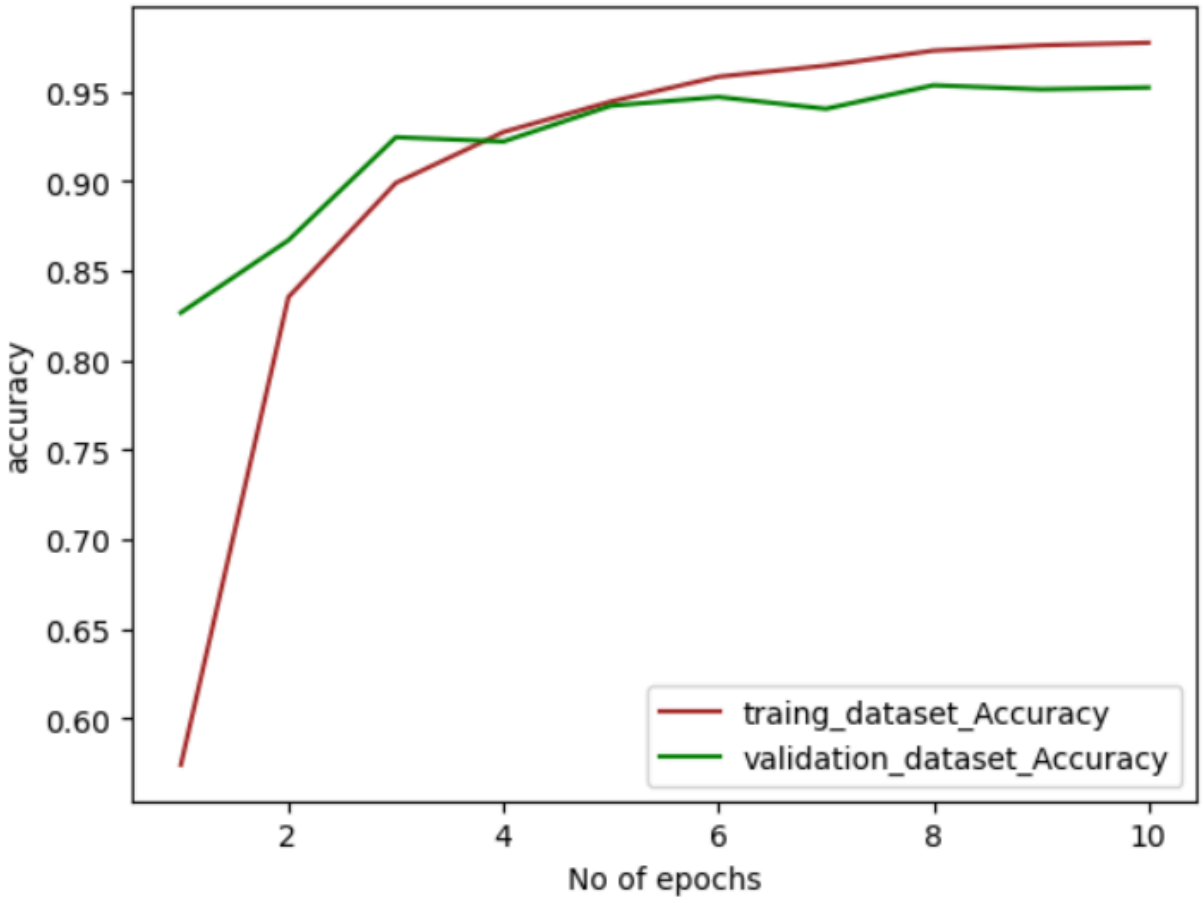
- **Final Layers:**

- Flatten layer to convert 2D feature maps into a 1D vector.

- Fully connected (Dense) layer with appropriate neurons for classification tasks (e.g., 38 output neurons for 38 disease classes).

- Output layer with softmax activation for multi-class classification.

Results:

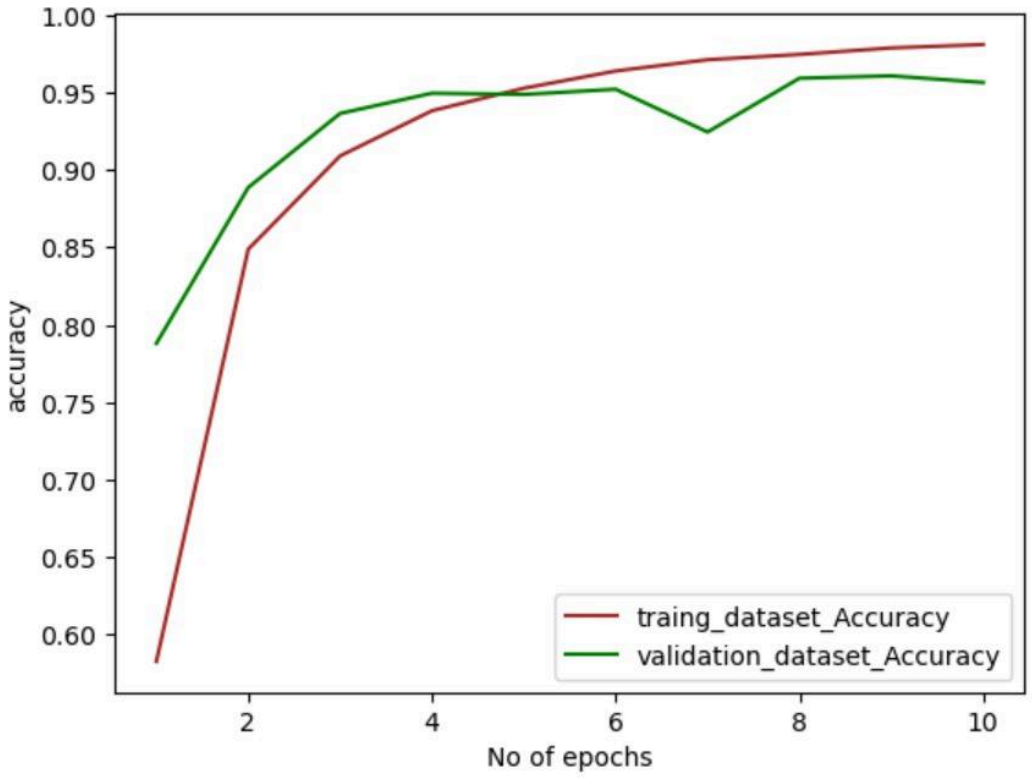


	precision	recall	f1-score	support
Apple___Apple_scab	0.98	0.90	0.94	504
Apple___Black_rot	1.00	0.95	0.97	497
Apple___Cedar_apple_rust	0.92	0.98	0.95	440
Apple___healthy	0.96	0.92	0.94	502
Blueberry___healthy	0.90	0.98	0.94	454
Cherry_(including_sour)___Powdery_mildew	0.97	0.98	0.97	421
Cherry_(including_sour)___healthy	0.99	0.98	0.99	456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.92	0.87	0.90	410
Corn_(maize)___Common_rust_	0.97	1.00	0.98	477
Corn_(maize)___Northern_Leaf_Blight	0.85	0.98	0.91	477
Corn_(maize)___healthy	0.99	0.99	0.99	465
Grape___Black_rot	0.93	0.93	0.93	472
Grape___Esca_(Black_Measles)	1.00	0.94	0.97	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0.94	0.99	0.96	430
Grape___healthy	0.99	0.99	0.99	423
Orange___Haunglongbing_(Citrus_greening)	0.99	0.99	0.99	503
Peach___Bacterial_spot	0.97	0.94	0.96	459
Peach___healthy	0.97	0.98	0.98	432
Pepper,_bell___Bacterial_spot	0.97	0.95	0.96	478
Pepper,_bell___healthy	0.94	0.94	0.94	497
Potato___Early_blight	0.95	0.99	0.97	485
Potato___Late_blight	0.93	0.95	0.94	485
Potato___healthy	0.94	0.94	0.94	456
Raspberry___healthy	0.92	1.00	0.95	445
Soybean___healthy	0.98	0.97	0.97	505
Squash___Powdery_mildew	0.97	0.99	0.98	434
Strawberry___Leaf_scorch	0.98	0.93	0.95	444
Strawberry___healthy	0.99	0.99	0.99	456
Tomato___Bacterial_spot	0.92	0.98	0.95	425
Tomato___Early_blight	0.93	0.86	0.89	480
Tomato___Late_blight	0.92	0.86	0.89	463
Tomato___Leaf_Mold	0.95	0.97	0.96	470
Tomato___Septoria_leaf_spot	0.93	0.83	0.87	436
Tomato___Spider_mites Two-spotted_spider_mite	0.95	0.92	0.93	435
Tomato___Target_Spot	0.85	0.92	0.89	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.99	0.99	0.99	490
Tomato___Tomato_mosaic_virus	0.98	0.98	0.98	448
Tomato___healthy	0.98	0.96	0.97	481
accuracy			0.95	17572
macro avg	0.95	0.95	0.95	17572
weighted avg	0.95	0.95	0.95	17572

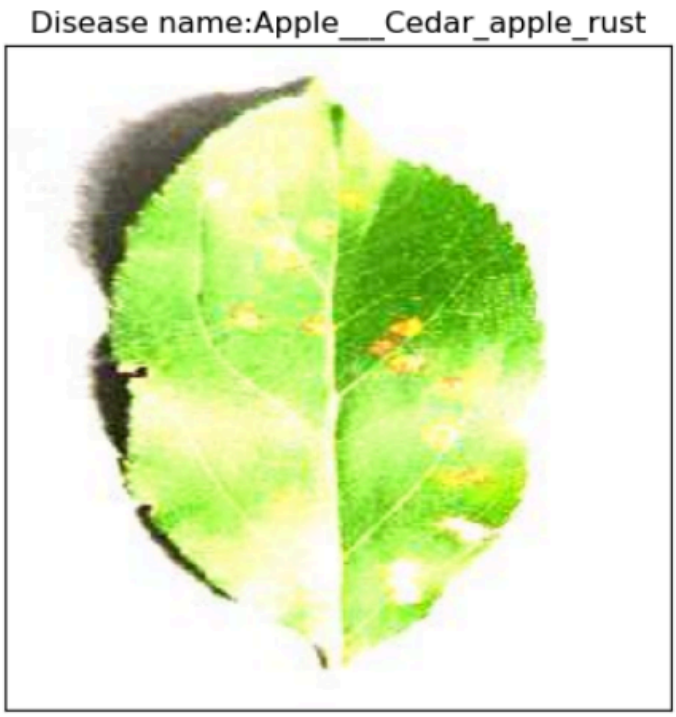
```
Epoch 1/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1805s 2s/step - accuracy: 0.3764 - loss: 2.2281 - val_accuracy: 0.8267 - val_loss: 0.5614
Epoch 2/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1405s 1s/step - accuracy: 0.8112 - loss: 0.5972 - val_accuracy: 0.8671 - val_loss: 0.4158
Epoch 3/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1390s 1s/step - accuracy: 0.8894 - loss: 0.3460 - val_accuracy: 0.9247 - val_loss: 0.2369
Epoch 4/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1403s 1s/step - accuracy: 0.9218 - loss: 0.2359 - val_accuracy: 0.9223 - val_loss: 0.2400
Epoch 5/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1420s 1s/step - accuracy: 0.9410 - loss: 0.1782 - val_accuracy: 0.9422 - val_loss: 0.1793
Epoch 6/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1398s 1s/step - accuracy: 0.9555 - loss: 0.1338 - val_accuracy: 0.9471 - val_loss: 0.1719
Epoch 7/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1406s 1s/step - accuracy: 0.9630 - loss: 0.1125 - val_accuracy: 0.9405 - val_loss: 0.1953
Epoch 8/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1409s 1s/step - accuracy: 0.9717 - loss: 0.0826 - val_accuracy: 0.9537 - val_loss: 0.1543
Epoch 9/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1408s 1s/step - accuracy: 0.9748 - loss: 0.0766 - val_accuracy: 0.9514 - val_loss: 0.1689
Epoch 10/10
1047/1047 ━━━━━━━━━━━━━━━━━━━━ 1416s 1s/step - accuracy: 0.9772 - loss: 0.0708 - val_accuracy: 0.9524 - val_loss: 0.1599
```

Comparison with different bootstrap dataset:

```
Epoch 1/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2676s 1s/step - accuracy: 0.3828 - loss: 2.1958 - val_accuracy: 0.7880 - val_loss:
0.6793
Epoch 2/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2930s 1s/step - accuracy: 0.8231 - loss: 0.5544 - val_accuracy: 0.8888 - val_loss:
0.3431
Epoch 3/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 1970s 897ms/step - accuracy: 0.8989 - loss: 0.3134 - val_accuracy: 0.9367 - val_los
s: 0.1980
Epoch 4/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 1935s 881ms/step - accuracy: 0.9338 - loss: 0.2079 - val_accuracy: 0.9497 - val_los
s: 0.1587
Epoch 5/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 1910s 869ms/step - accuracy: 0.9490 - loss: 0.1559 - val_accuracy: 0.9490 - val_los
s: 0.1606
Epoch 6/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2236s 1s/step - accuracy: 0.9625 - loss: 0.1149 - val_accuracy: 0.9523 - val_loss:
0.1590
Epoch 7/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2181s 993ms/step - accuracy: 0.9692 - loss: 0.0934 - val_accuracy: 0.9247 - val_los
s: 0.2356
Epoch 8/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2090s 951ms/step - accuracy: 0.9723 - loss: 0.0854 - val_accuracy: 0.9594 - val_los
s: 0.1405
Epoch 9/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2034s 926ms/step - accuracy: 0.9780 - loss: 0.0645 - val_accuracy: 0.9610 - val_los
s: 0.1282
Epoch 10/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 2142s 975ms/step - accuracy: 0.9815 - loss: 0.0564 - val_accuracy: 0.9567 - val_los
s: 0.1438
```



Prediction of the model on test set:




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
(array([[8.43180980e-15, 3.76072464e-15, 1.00000000e+00, 1.34924136e-13,
         2.55859001e-09, 1.17245935e-11, 5.65132566e-18, 2.22677610e-14,
         1.76849428e-17, 2.16493109e-15, 7.36863398e-19, 1.34462677e-14,
         3.32554622e-14, 4.02370773e-16, 8.85734213e-17, 1.35414707e-10,
         1.24785746e-11, 7.41692965e-15, 2.82380456e-13, 1.70687759e-12,
         2.96994070e-17, 1.03212834e-17, 4.41417710e-15, 7.98830538e-14,
         7.90675147e-18, 4.76758371e-16, 1.10404847e-15, 2.09643471e-15,
         1.55192019e-08, 5.44905579e-11, 2.61304589e-10, 2.12492415e-12,
         1.43705686e-10, 7.77586622e-16, 2.42850948e-11, 7.14810744e-09,
         3.64415303e-10, 1.19772991e-14]], dtype=float32),
 (1, 38))
```

Resources:

https://www.tensorflow.org/

https://www.tensorflow.org/guide/keras

https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

https://seaborn.pydata.org/

https://ieeexplore.ieee.org/document/8437085