Volume 9, Issue 7, July – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUL195

Using Machine Learning to Identify Diseases and Perform Sorting in Apple Fruit

Arpit Patidar[1], Abir Chakravorty[1*]
[1]Agricultural and Food Engineering Department, IIT Kharagpur
*Orc Id: https://orcid.org/0000-0002-9253-2594

Abstract:- Fruit diseases play a major role in global agriculture, leading to substantial crop losses and influencing food production and economic stability. In this age of Industry 4.0 the fruit sorting is an important part in the food processing wherein this work plays a vital role. In this study, a solution for the detection and classification of apple fruit diseases is proposed and experimentally validated. Deep learning models offer promise for automating disease identification using fruit images, but encounter obstacles such as the requirement for extensive training data, computational complexity, and the risk of overfitting. This study introduces an innovative convolutional neural network (CNN) architecture aimed at addressing these challenges by incorporating a reduced number of layers, thus alleviating computational burdens while maintaining performance. Additionally, augmentation techniques such as shift, shear, scaling, zoom, and flipping are employed to diversify the training set without additional image acquisition. Our CNN model is specifically trained to identify common apple crop diseases like Scab, Rot, and Blotch. Rigorous experimental evaluation demonstrates the effectiveness of our model, achieving a remarkable classification accuracy of 95.37%. Significantly, our model demonstrates reduced storage requirements and faster execution times compared to existing deep CNN architectures, enabling deployment on handheld devices and resource-limited environments. While other CNN models may offer similar accuracy levels, our approach emphasizes efficiency and resource optimization, rendering it practical for real-world applications in agriculture. Furthermore, our CNN model exhibits resilience to environmental variations and imaging parameters, enhancing its applicability across diverse agricultural settings. By leveraging advanced machine learning techniques, the approach developed in this experimental work contributes to modernizing fruits and vegetables sorting operations in food processing, crop management practices thus promoting agricultural sustainability. The scalability and portability of our model make it suitable for deployment in both small-scale farms and large-scale agricultural operations.

Keywords: Apple diseases, classification, convolutional neural network, deep learning, disease detection, image processing, machine learning, fruit sorting, automation.

I. INTRODUCTION

In India, while the apple cultivation area has expanded by 20%, production has only seen amodest increase of 1-2% (National Horticulture Board 2020). Pests and diseases pose significant challenges to apple crop production worldwide, with fungal diseases particularly impacting fruit quality in regions like Himachal Pradesh, India's second-largest producer of apples. Plant infections are broadly categorized as biotic andabiotic, with pathogens such as viruses, fungi, and bacteria responsible for biotic diseases (V. K. Vishnoi et al. 2021). Biotic diseases, being highly transmissible and hazardous, pose greater risks compared to abiotic diseases caused by factors like mineral deficiency or environmental stressors.

➢ *Introduction of Apple Disease Detection*

Apple fruits are frequently affected by biotic diseases such as scab, cedar rust, blotch, powdery mildew, blight, mosaic, and black rot. It is important to identify scab, blotch, and black rot. Traditional disease detection methods rely on visual observation by experts, which can be costly and time-consuming, particularly in remote regions. Since diseases can result in significant losses in both yield and quality at harvest, it becomes essential to identify symptoms early in the fruit development stages through automatic disease detection. Identifying diseases promptly aids in planning control measures for the following year to mitigate losses. Additionally, some diseases spread to the tree's branches, twigs, and fruit. Scratching, rot, and blotch are common diseases of apples. When an apple rot infection occurs, the affected area becomes circular and brown or black, with a red halo around it. On the other hand, apple scabs appear as gray or brown corky spots. A fungus called apple blotch causes dark, asymmetrical, or lobed lesions on the fruit's surface.

While machine vision systems are currently used in the industry to automatically inspect apples for size and color, it is still difficult to identify flaws in apples. This challenge is caused by the inherent differences in skin tone between apple varieties, the variety of defect kinds, and the existence of calyxes and stems. Many existing methods for apple defect segmentation rely on simple thresholding approaches. For instance, a globally adaptive thresholding method, based on a modified version of Otsu's algorithm, was proposed for segmenting fecal contamination defects on apples. Classification-based techniques aim to categorize pixels into different classes using various classification methods.

Bayesian classification is commonly utilized, where pixels are compared against a pre-calculated model and classified as either defective or healthy. Unsupervised classification, on the other hand, lacks target values and thus provides no guidance during the learning process. Some researchers have employed unsupervised classification for defect segmentation purposes. In another approach, (Ojala et al. 2002) utilized the Local Binary Pattern (LBP) histogram for rotation-invariant texture classification, demonstrating impressive outcomes on representative texture databases. LBP has also found applications in face recognition, dynamic texture recognition, and shape localization. Convolutional neural networks (CNNs) have been proposed as a comprehensive modeling approach for pattern recognition tasks. I have proposed and experimentally validated the effectiveness of using CNNs as classifiers for the automatic detection and classification of fruit diseases. To validate our approach, I focused on three common apple diseases: blotch, rot, and scab.

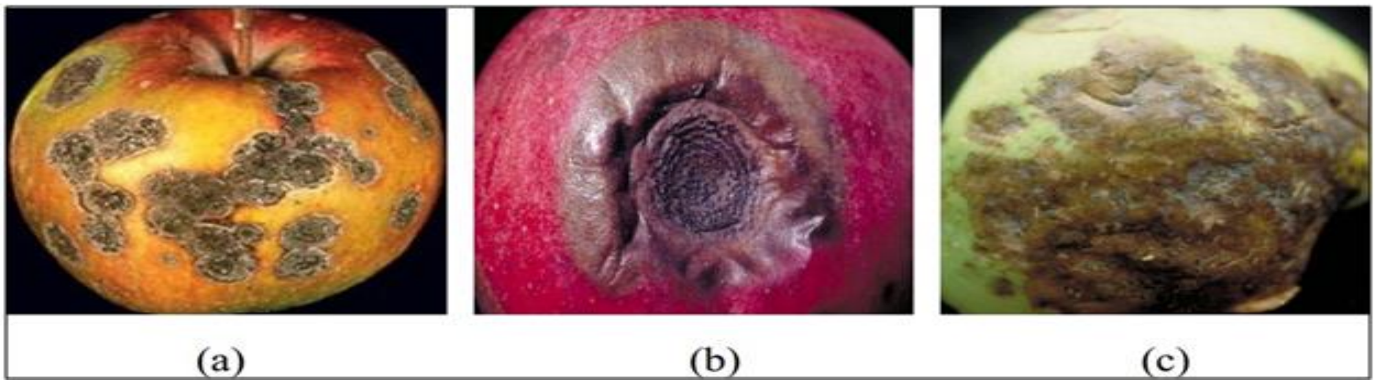


Fig 1 Three Common Apple Diseases. a) Scab, (b) Rot and, (c) Blotch

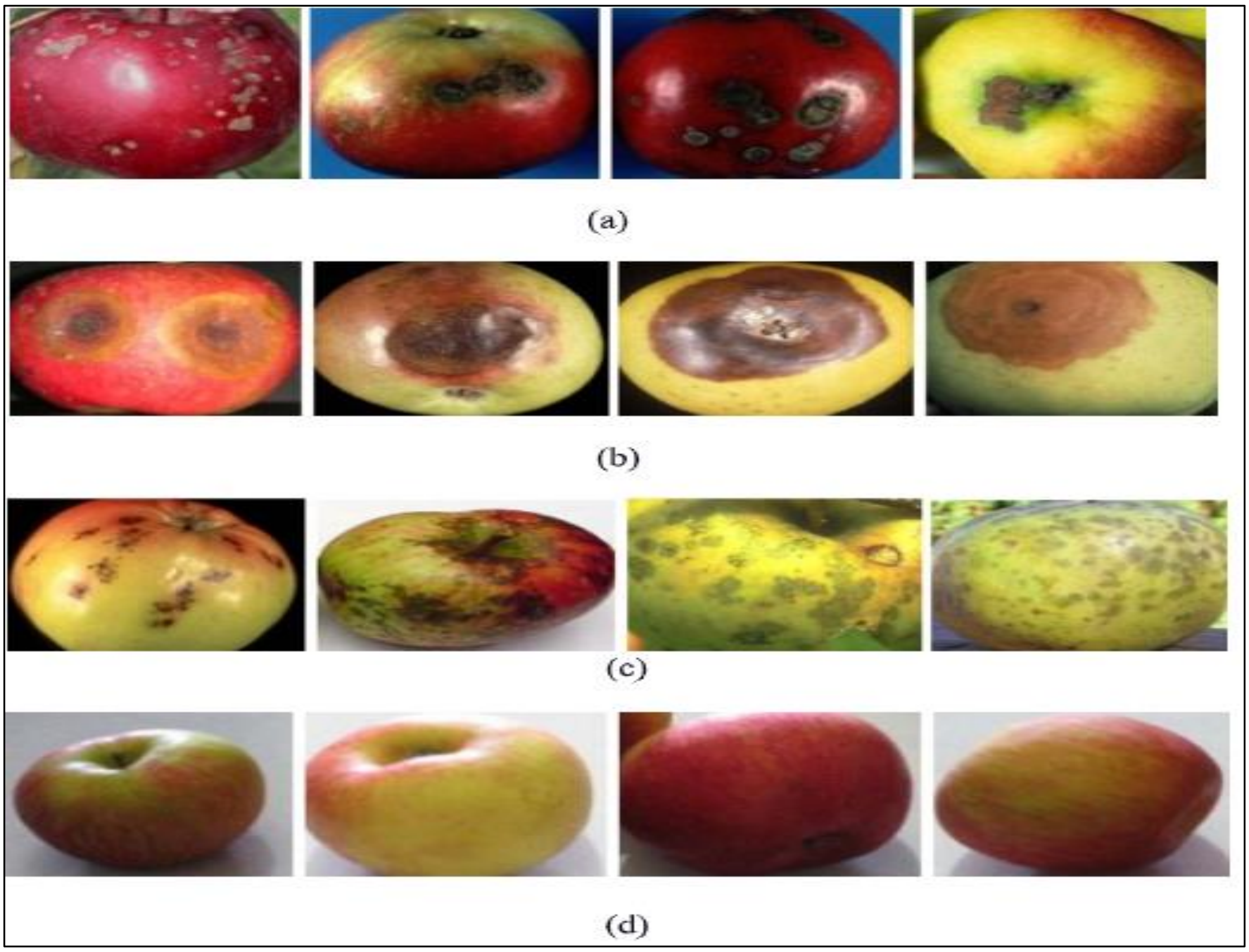


Fig 2 Sample Images from the Data Set of Type (a) Apple Scab (b) Apple Rot (c) Apple Blotch, and (d) Normal Apple

Machine Learning (ML) techniques such as k-nearest neighbor (KNN), artificial neural networks (ANN), and support vector machine (SVM) are commonly employed for fruit disease detection and identification. However, deep learning methods have emerged as amore effective approach for hierarchical feature learning, capable of representing both low-level and high-level features with increased complexity. Convolutional neural networks, or CNNs, have proven effective in a number of applications, such as speech recognition, object detection, image classification, and natural language processing. CNNs' effectiveness in identifying fruit diseases has also been demonstrated. However, deploying CNN models on portable devices like smartphones, which are essential for supporting farmers becomes difficult due to the computational demands of processing large datasets. By creating a lightweight deep CNN model that reduces computational demands and effectively diagnoses illnesses, this work attempts to address this challenge. I present a novel four-convolutional-layer deep CNN model (Conv4 DCNN) that is specifically intended to use fruit images for the diagnosis of three different apple fruit diseases. Techniques for augmenting data are used to improve model performance, and the random search technique is used to optimize hyperparameters and find the best possible values. This work's main contribution is the creation of a low-complexity deep neural network model specifically designed to identify apple fruit diseases with minimalcomputational burden.

➢ *Importance of Apple Fruit Disease Detection*

The early detection and timely management of diseases in apple orchards are crucial for ensuring sustainable production and minimizing economic losses. Traditionally, disease detection has relied on visual inspection by agricultural experts, which islabor-intensive, subjective, and prone to errors. Moreover, the delay in disease diagnosiscan lead to ineffective control measures, resulting in reduced yields and compromisedfruit quality.

➢ *Role of Machine Learning*

The advent of machine learning techniques has revolutionized disease detection in agriculture by providing automated and accurate methods for identifying fruit diseases. Machine learning algorithms, particularly those based on deep learning, have shown remarkable performance in analyzing large datasets, including images, and extracting meaningful patterns. By leveraging these algorithms, it becomes possible to develop efficient and reliable systems for detecting apple fruit diseases at an early stage.

➢ *Objectives*

The primary objective of this research is to develop a machine learning-based approach for the detection of apple fruit diseases. Specifically, the study aims to:

- To investigate the feasibility of using machine learning algorithms for identifying common diseases affecting apple fruits.
- To develop and optimize machine learning models capable of classifying diseasedand healthy apple fruits based on image data.
- To evaluate the performance of the developed models in terms of accuracy, sensitivity, specificity, and computational efficiency.
- To explore the potential applications of the proposed approach in real-world agricultural settings and its implications for crop management and disease control.

II. LITERATURE REVIEW

➢ *Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition*

De Luna et al. (2019) introduced an innovative method to enhance agricultural productivity, specifically in tomato cultivation, by integrating smart farming systems. Their research addresses the challenge of detecting diseases in tomato plants by utilizing recent advancements in computer vision, particularly deep learning techniques. They developed a motor-controlled image capturing system capable of photographing every side of tomato plants to facilitate disease identification and recognition. The study focused on the Diamante Max tomato variety and aimed to detect three commondiseases: Phroma Rot, Leaf Miner, and Target Spot. To classify tomato diseases, a deep convolutional neural network (CNN) was trained using a dataset containing both diseased and healthy plant leaves. The Transfer Learning disease recognition model achieved an accuracy of 95.75%, while the F-RCNN anomaly detection model scored 80% confidence. Additionally, the automated image capture system demonstrated a 91.67% accuracy rate in identifying diseases on tomato plant leaves.

➢ *CNN based Leaf Disease Identification and Remedy Recommendation System*

Rohan et al. (2019) presented a novel approach named CNN based Leaf Disease Identification and Remedy Recommendation System. They underscored the vital role of agriculture in the econoour and society, highlighting the challenges faced by farmers in identifying leaf diseases, which often lead to reduced crop yields. The authors stressed the potential of utilizing leaf videos and images to provide valuable insights for effective disease management and increased productivity. Early detection and treatment of diseases are crucial to minimize negative impacts on harvest yields and ensure adequate crop nutrition. The emergence of smart devices capable of identifying anddetecting plant diseases has facilitated more efficient responses to disease outbreaks in the agriculture sector. Leveraging publicly available datasets containing 5,000 photos of both healthy and diseased plant leaves, the study focused on plant disease detection through image processing techniques. Crop types were classified, and diseases were categorized into four classes using semi-supervised techniques.

➢ *Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolution Neural Networks*

Liu et al. (2019) focus on the identification of common apple leaf diseases affecting apple crops, such as aria leaf spot, brown spot, mosaic, grey spot, and rust. They employ deep learning methods, particularly enhanced convolutional neural networks (CNNs), to detect these diseases. The study utilizes the Apple Leaf Disease Dataset (ALDD), which includes both intricate and lab-created images of apple leaves. To enhance the dataset, the researchers utilize data augmentation and image annotation technologies, facilitating the development of a new deep CNN-based model for disease detection. This model incorporates the GoogleNet Inception structure and Rainbow concatenation. Through training and evaluation on a dataset comprising 26,377 disease images, the proposed INAR-model achieves a high detection speed of 23.13 frames per second (FPS) with a detection performance of 78.80%. The INAR-SSD model demonstrates a promising method for early detection of apple leaf diseases, offering real-time detection capabilities with enhanced accuracy and speed compared to previoustechniques.

➢ *Identification of plant leaf diseases using a nine-layer deep convolutional neural network*

Geetharamani and Arun Pandian (2020) propose a method for identifying plant leaf diseases employing deep convolutional neural networks (CNNs). Their approach involves utilizing a dataset consisting of background images and more than 39 distinct classes of plant leaf diseases to train the CNN model. To enhance the dataset's quality, they employ six data augmentation techniques: gamma correction, image flipping, principal component analysis (PCA), color augmentation, rotation, noise injection, and scaling. Through experimentation, the authors find that integrating data augmentation techniques significantly improves the model's performance. They explore various combinations of training epochs, batch sizes, and dropout rates to optimize the model training process. Comparative analysis with transfer learning techniques indicates that the proposed CNN model outperforms them, especially when assessed with independentdata. The CNN model achieves a classification accuracy of 96.46% through simulations, surpassing the performance of transfer learning techniques. Overall, the study demonstrates the superior accuracy of the CNN model compared to transfer learning methods in identifying plant leaf diseases.

➢ *Crop Disease Detection Using Deep Learning*

Kulkarni (2018) addresses the rising prevalence of crop diseases attributed to recent climate changes and crops' susceptibility to various pathogens. This phenomenon leads to financial losses for farmers, widespread crop destruction, and reduced agricultural productivity. Identifying and treating these diseases pose significant challenges due to the rapid proliferation of diverse diseases and limited knowledge among farmers. The task is further complicated by the similarities in texture and appearance among diseased and healthy leaves. To tackle this issue, Kulkarni proposes a deep learning-based model leveraging computer vision techniques. The model is trained on a dataset comprising images of crop leaves under both healthy and diseased conditions. Through this approach, the model effectively classifies leaf images into diseased categories based on the pattern of defects.

➢ *Identification of Plant Disease using Image Processing Technique*

Rathan et al. (2019) address the evolving role of agriculture, emphasizing its significance in sustaining populations, particularly in regions where a substantial portion of the population relies on agriculture for livelihoods. However, crop diseases frequently undermine agricultural productivity, leading to significant losses. Early detection of leaf diseases is crucial for mitigating such losses. The paper aims to develop software capable of automatically identifying and categorizing plant diseases. The process involves steps such as image acquisition, preprocessing, segmentation, feature extraction, and classification. Leaf images are utilized for disease identification, highlighting the importance of employing image processing techniques in the identification and categorization of agricultural diseases.

➢ *GUI based Detection of Unhealthy Leaves using Image Processing Techniques*

Vamsi et al. (2019) emphasize the critical role of enhancing agricultural productivity for the growth of the Indian econoour. They propose the utilization of image processingtechniques to identify unhealthy leaves, aiming to facilitate the development of an intelligent and efficient farming system. Focusing on ladies finger plant leaves, thestudy aims to detect early signs of diseases such as powdery mildew, leaf spot, andyellow mosaic vein. The process involves capturing leaf images, preprocessing, segmenting, extracting features, and classifying them. Additionally, the study accounts for practical challenges such as noisy image datasets, terrain conditions, and climate variations. For classification, Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are employed, while K-Means clustering is utilized for segmentation. Principal Component Analysis (PCA) is applied to reduce the feature set. Experimental results demonstrate that SVM and ANN achieve average disease detection accuracies of85% and 97%, respectively. In noise-free conditions, these accuracies improve to 92% and 98%, respectively. The study lays the foundation for advancing automation in the agriculture sector.

➢ *Automatic Disease Symptoms Segmentation Optimized for Dissimilarity Feature Extraction in Digital Photographs of Plant Leaves*

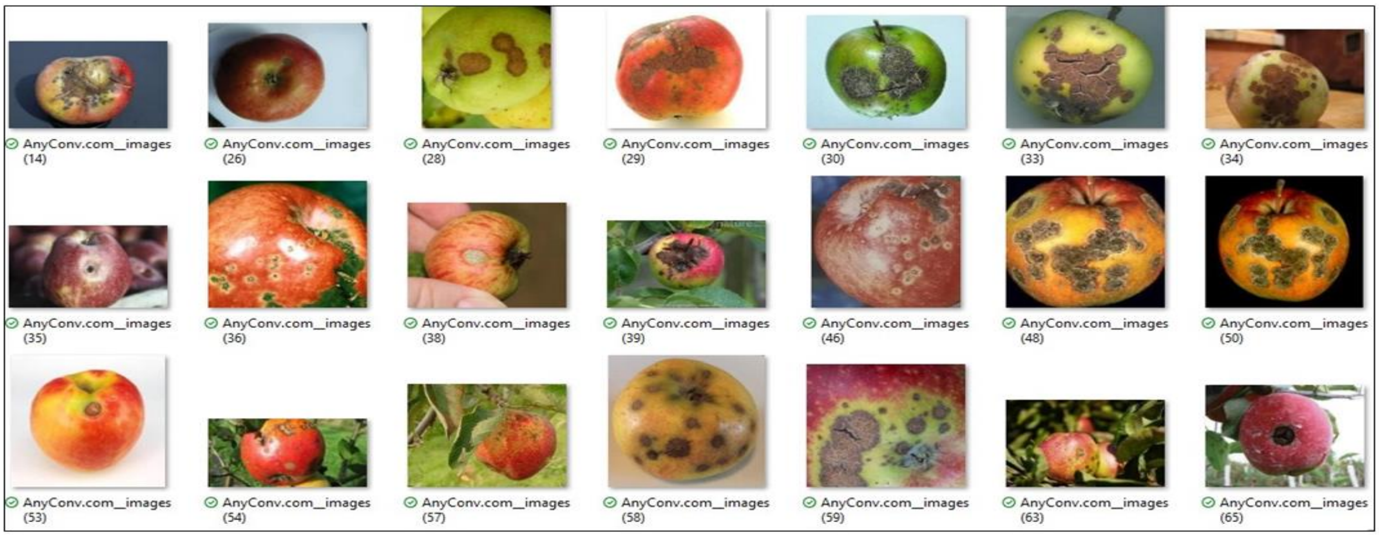
Abdu et al. (2017) address the critical initial step of applying machine learning to plant disease detection, which involves segmenting diseased symptom regions in plant leaf images. This process, referred to as Region of Interest (ROI) segmentation, aims to isolate areas exhibiting symptoms with color variations from the surrounding green tissue. These segmented regions serve as the basis for extracting discriminant features. However, existing approaches often fall short in capturing the comprehensive evolution of disease symptoms from onset to complete manifestation, which could provide valuable dissimilarity features for disease characterization. Moreover, challenges arise from the structure of the disease itself and the circumstances during image capture, complicating traditional ROI segmentation techniques. For example, disease symptoms may blend into healthy green tissue, blurring the separation line and potentially leading to inaccuracies. To overcome these limitations, the study proposes an automated extended Region of Interest (EROI) segmentation technique. This method utilizes color homogeneity thresholding to expand the border region, incorporating portions of healthy tissue to provide insights into symptom progression. Both standard ROI segmentation and a reduced ROI approach were applied to a well-known Plant Village dataset to validate the proposed methodology.

III. MATERIALS AND METHODS

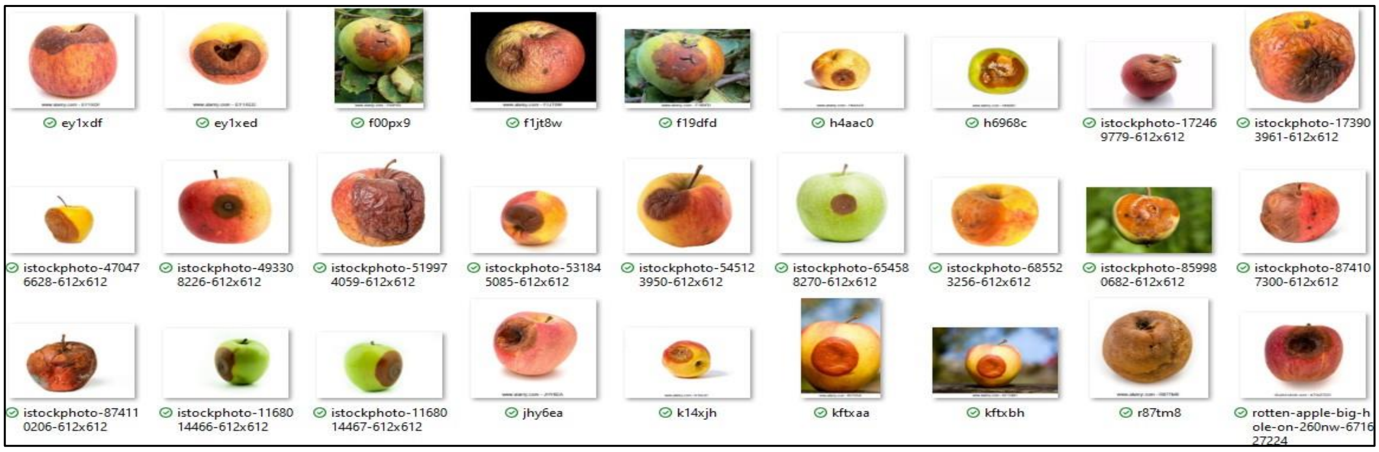
This section covers complete details related to the development of deep CNN (Conv-4 DCNN) model to investigate the diseases in apple crops. The process of training and validating the proposed deep CNN model is presented along with the detailed mathematical formulations.

➢ *Dataset Details*

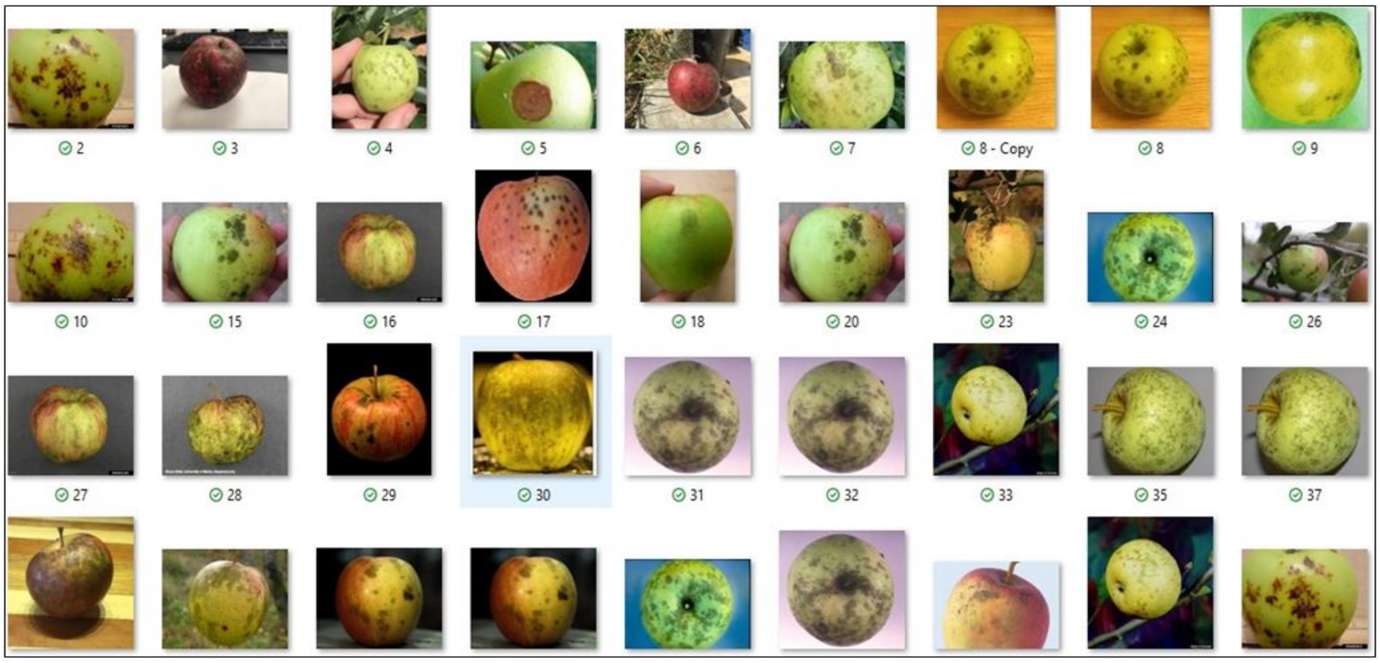
The apple images used in this study were captured from the Machua fruit mandi in Kolkata and some data were taken from publicly available on Kaggle (Shiv Ram Dubey Et al. 2012). The dataset comprises a total of 5915 RGB images of apple fruits, categorized into four classes. These classes correspond to three common apple diseases: blotch, scab, and apple rot. Additionally, there is a fourth class representing healthy/uninfected apple images. Sample images from each class are illustrated in Figure 3.1. The naming convention for the disease classes aligns with the specific apple diseases, while the healthy class represents images of fruit without any disease. The diversity of images within the dataset is crucial for effective training, allowing the model to learn important variations. The deep CNN model's generalizability is improved by this diversity. Several image transformations, including shift, shear, scaling, zoom, and flipping, are used to expand the dataset and add more variations (Peng jiang 2019). These adjustments produce subtle changes to the images, adding a variety of examples to the training set. Thus, by enhancing the model's ability to generalize and reduce overfitting, this augmentation technique enhances the model's performance and tolerance.



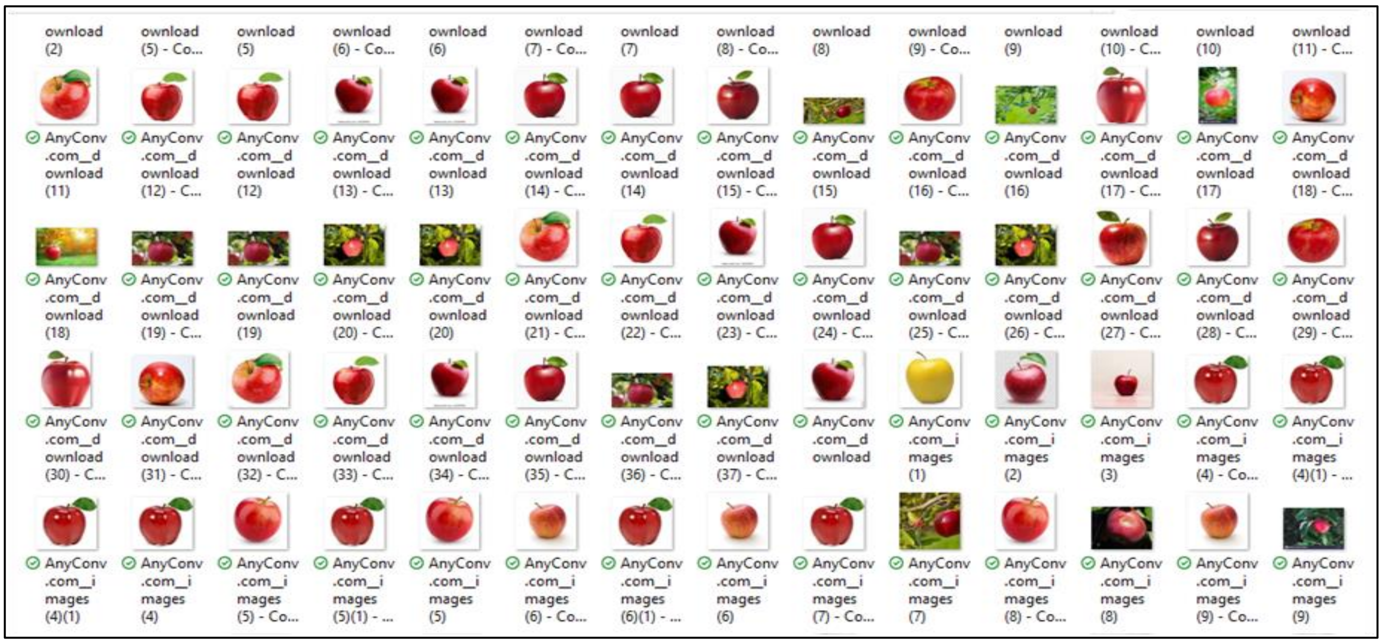
a. Apple Scab Dataset


ey1xdf
ey1xed
f00px9
f1jt8w
f19dfd
h4aac0
h6968c
istockphoto-17246 9779-612x612
istockphoto-17390 3961-612x612
istockphoto-47047 6628-612x612
istockphoto-49330 8226-612x612
istockphoto-51997 4059-612x612
istockphoto-53184 5085-612x612
istockphoto-54512 3950-612x612
istockphoto-65458 8270-612x612
istockphoto-68552 3256-612x612
istockphoto-85998 0682-612x612
istockphoto-87410 7300-612x612
istockphoto-87411 0206-612x612
istockphoto-11680 14466-612x612
istockphoto-11680 14467-612x612
jhy6ea
k14xjh
kftxaa
kftxbh
r87tm8
rotten-apple-big-hole-on-260nw-6716 27224

b. Apple rot Dataset



c. Apple Blotch Dataset


ownload (2)
ownload (5) - Co...
ownload (5)
ownload (6) - Co...
ownload (6)
ownload (7) - Co...
ownload (7)
ownload (8) - Co...
ownload (8)
ownload (9) - Co...
ownload (9)
ownload (10) - C...
ownload (10)
ownload (11) - C...
AnyConv.com_download (11)
AnyConv.com_download (12) - C...
AnyConv.com_download (12)
AnyConv.com_download (13) - C...
AnyConv.com_download (13)
AnyConv.com_download (14) - C...
AnyConv.com_download (14)
AnyConv.com_download (15) - C...
AnyConv.com_download (15)
AnyConv.com_download (16) - C...
AnyConv.com_download (16)
AnyConv.com_download (17) - C...
AnyConv.com_download (17)
AnyConv.com_download (18) - C...
AnyConv.com_download (18)
AnyConv.com_download (19) - C...
AnyConv.com_download (19)
AnyConv.com_download (20) - C...
AnyConv.com_download (20)
AnyConv.com_download (21) - C...
AnyConv.com_download (22) - C...
AnyConv.com_download (23) - C...
AnyConv.com_download (24) - C...
AnyConv.com_download (25) - C...
AnyConv.com_download (26) - C...
AnyConv.com_download (27) - C...
AnyConv.com_download (28) - C...
AnyConv.com_download (29) - C...
AnyConv.com_download (30) - C...
AnyConv.com_download (31) - C...
AnyConv.com_download (32) - C...
AnyConv.com_download (33) - C...
AnyConv.com_download (34) - C...
AnyConv.com_download (35) - C...
AnyConv.com_download (36) - C...
AnyConv.com_download (37) - C...
AnyConv.com_download
AnyConv.com_images (1)
AnyConv.com_images (2)
AnyConv.com_images (3)
AnyConv.com_images (4) - Co...
AnyConv.com_images (4)(1) - ...
AnyConv.com_images (4)(1)
AnyConv.com_images (4)
AnyConv.com_images (5) - Co...
AnyConv.com_images (5)(1) - ...
AnyConv.com_images (5)
AnyConv.com_images (6) - Co...
AnyConv.com_images (6)(1) - ...
AnyConv.com_images (6)
AnyConv.com_images (7) - Co...
AnyConv.com_images (7)
AnyConv.com_images (8) - Co...
AnyConv.com_images (8)
AnyConv.com_images (9) - Co...
AnyConv.com_images (9)

d. Normal Apple Dataset
Fig 3 Apple Dataset

➢ *Implementation Work*

A deep Convolutional Neural Network (CNN) is an architecture of a feed-forward Artificial Neural Network (ANN) commonly utilized in deep learning (Vihaor Kumar Vishnoi et al. 2023). In CNNs, the term 'deep' refers to the incorporation of numerous layers, unlike traditional neural networks. Typically, a deep CNN is built by stacking various layers, including convolutional layers with non-linear activation functions, pooling layers, and fully connected layers. Deep CNNs offer numerous advantages over traditional machine learning methods, notably in their ability to automatically extract features without the need for manual feature engineering. The initial layers of a deep CNN are responsible for capturing basic, low-level features from the input data. Figure 4 illustrates the process of data manipulation.

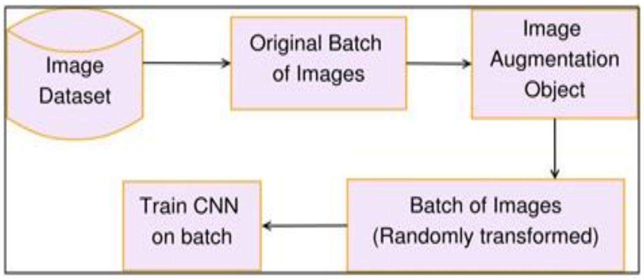


Image Dataset
Original Batch of Images
Image Augmentation Object
Batch of Images (Randomly transformed)
Train CNN on batch

Fig 4 Representation of the Process of Data Augmentation/Manipulation

- *Convolutional Layer*

Convolution is carried out by the convolutional layer by applying a filter (kernel) to an input image. By locating regional patterns found in the layers before, it creates feature maps. The linear convolution operation and the non-linear activation function are the two primary components of the convolutional layer. Equation (1) illustrates how this operation, which convolves the filter over multi-channel image volumes like RGBimages, is expressed.

$$\mathrm{Conv}\,(I,\ K)x,\ y = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k}\ I_{x+i-1,y+j-1,k} \text{ -----------------------} \quad (1)$$

In the convolution process, the kernel K (f_h, f_w, n_C) convolves with the image I (n_H, n_W, n_C) of similar channel numbers nC but potentially different sizes, resulting in the generation of a feature map O (o_H, o_W, z). Here, f_h and f_w represent the height and width of the kernel respectively, while nH and nW denote the height and width of the input image. Typically, the kernel is traditionally regarded as an odd-dimensional square window, with $f_H = f_w =$ f.

$$\text{Feature\_map}\,(o_H, o_W, z) = \left(\left[(n_H + 2p - f)/s\ +\ 1\right],\ \left[(n_W + 2p - f)/s\ +\ 1\right],\ z\right) \text{-------} (2)$$

In the given equation 2, the symbol p represents the padding value, s denotes the stride, and z signifies the number of kernels convolved with the input image.

The Rectified Linear Unit (ReLU) stands as one of the most commonly used activation functions. ReLU operates in a manner where it does not activate all neurons simultaneously. Neurons are activated only if the output of the convolution unit or otherlinear transformation is equal to or greater than zero. This activation function is represented as shown in equation (3):

$$f\,(z) = \max(0,z) \text{...} (3)$$

- *Pooling Layer*

The feature maps created by the convolutional layers are downsampled by the pooling layer. Its goal is to reduce the size of activation maps that might have too many parameters. As such, it reduces processing overhead, lessens overfitting, and shortens training times. Max, min, and average pooling are three of the main pooling techniques. But the most common method is max pooling, which takes the maximum value out of every input patch. The following equation (3) shows the max pooling operation:

$$\text{Max\_Pooling}: y_j = \max_{i \in R_j} (P_i) \qquad (4)$$

Where R denotes a receptive field containing P pixels. The dimension of the generatedfeature map is defined in (4).

$$\text{Feature\_map}\,(o_H, o_W, n_C) = \left(\left[(n_H + 2p - f)/s\ +\ 1\right],\ \left[(n_W + 2p - f)/s\ +\ 1\right],\ n_C \text{ ---} (5)$$

- *Fully Connected Layer*

In order to create output layers with a predetermined number of outputs, the fully connected (FC) or dense layers in a CNN are usually positioned toward the end of the CNN architecture. These layers bear similarities to those found in traditional neural networks. The data that FC layers work with is one-dimensional (1-D). The flatten layer comes before the FC layers and transforms the two-dimensional (2-D) output from earlier layers into a one-dimensional (1-D) representation. The transformations carried out by the FC layers are both linear and non-linear, and they can be mathematically represented as follows in equations (6) - (7):

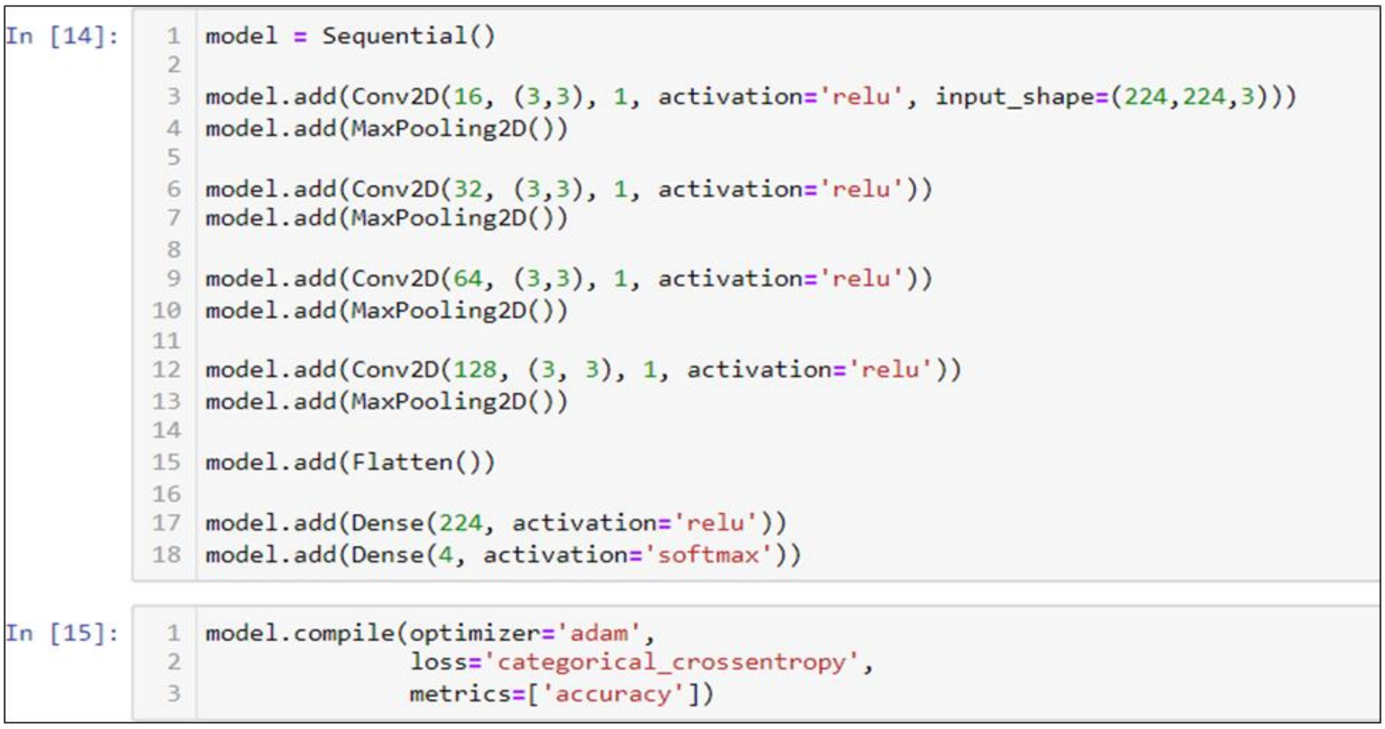
$$Z = W^{T*}\,X + b \underline{\qquad\qquad\qquad\qquad\qquad\qquad} (6)$$

$$O = f\,(Z) \underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad} (7)$$

In the provided equations, X represents the input feature map, W denotes the weight, b signifies the bias terms, and O represents the output of the fully connected layer. To enhance prediction accuracy, optimal weights are essential to minimize the loss function. The gradient descent method is commonly employed to determine these optimal weights. Another technique, the Adam algorithm, offers a smoother and less noisy optimization process by adjusting the learning rate based on adaptive estimates of lower-order moments. This facilitates learning rate annealing during gradient optimization.

- *Proposed Method*

The proposed Conv-4 DCNN model comprises four convolutional layers and two fully connected layers following three max-pooling units. Rectified Linear Unit (ReLU) serves as the non-linear activation function in each convolutional layer and the initial dense layer. For classifying apple fruit diseases, the softmax function is applied at the output layer. This function facilitates multiclass classification by assuming that each sample belongs to precisely one class. Figure 5 illustrates the sequencing of different layers alongside the activation functions integrated into the developed deep CNNmodel. Furthermore, the dropout unit functions by randomly eliminating certain neurons that rely on any one feature, therefore some neurons are ignored to spread out the weights for better generalization. Initially, at the first convolution level 16 filters (3 × 3) with valid padding and stride (1, 1) are selected to convolute over RGB images of size 256 × 256.



```
In [14]:  model = Sequential()

          model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(224,224,3)))
          model.add(MaxPooling2D())

          model.add(Conv2D(32, (3,3), 1, activation='relu'))
          model.add(MaxPooling2D())

          model.add(Conv2D(64, (3,3), 1, activation='relu'))
          model.add(MaxPooling2D())

          model.add(Conv2D(128, (3, 3), 1, activation='relu'))
          model.add(MaxPooling2D())

          model.add(Flatten())

          model.add(Dense(224, activation='relu'))
          model.add(Dense(4, activation='softmax'))

In [15]:  model.compile(optimizer='adam',
                        loss='categorical_crossentropy',
                        metrics=['accuracy'])
```

Fig 5 Pipelining of Layers in Deep CNN Model

- *Model Building*

The deep CNN model is evaluated and trained using the dataset that is covered in Section 3.1. In a 60-30-10 ratio, the training, validation, and test sets are divided into this dataset using the hold-out validation technique. That is why the training, validation, and test sets consist of 3549, 1775, and 591 images, respectively, as Fig 6 shows the deep CNN model.

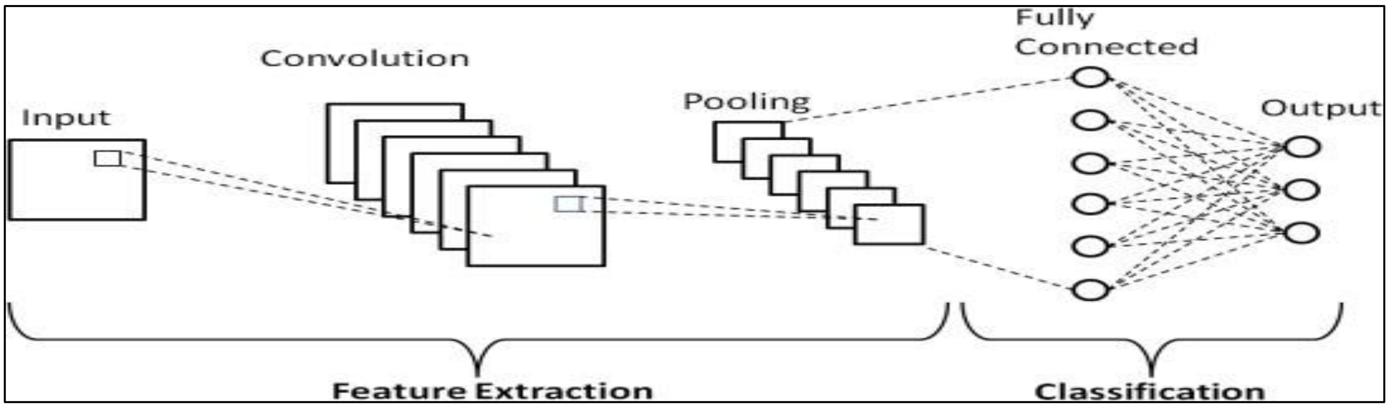


Fig 6 Deep CNN model

The deep CNN model is built in Python on the Jupyter Notebook virtual platform, which has an Nvidia P100 TPU and 12 GB of RAM, using the Keras, Scikit-learn libraries. The Matplotlib library is used to display the outcomes for visualization. The model is trained with a minibatch size of 56 and a learning rate of 0.0001 over a span of 20 epochs. Equation (8) defines categorical cross-entropy, which is the loss function of choice because it works well for multiclass classification tasks.

$$\text{loss(L)} = -\sum_{m=1}^{N} y_{i,m} \, log(p_{i,m}) \text{ ----------------------------------- } (8)$$

In the given equation, p is the probability of correctly predicting that observation i belongs to class m, N is the number of classes (where N > 2), and y is a binary indicator indicating the correct labeling of class m for observation i. The Adam optimization algorithm is used for gradient optimization, with the first and second moments' exponential decay rates set at β1 = 0.9 and β2 = 0.999, respectively. To find the best hyperparameters, random search techniques are used for hyperparameter tuning. Moreover, the feature map produced by the second convolutional layer offers more precise information about the lesions connected to a specific disease and less general information about the image.

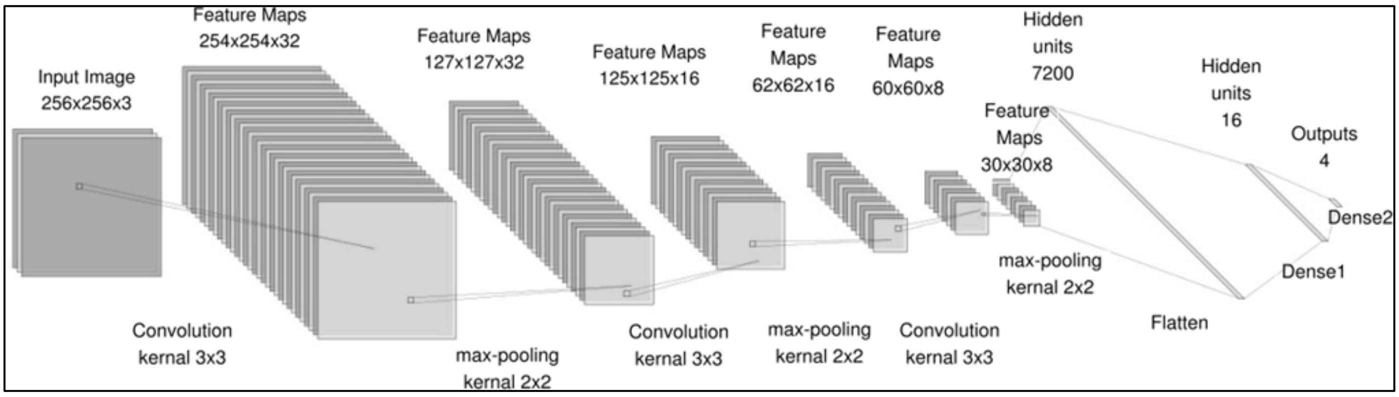


Fig 7 Layered architecture of proposed deep CNN model for apple disease detection

✓ *Image Acquisition:*

Image acquisition in the first load the image in a digital picture process that consists of capturing the image through a digital camera and stores it in digital media for additional python operations (Vibhor Kumar Vishnoi et al. 2022).

✓ *Image Preprocessing:*

The main aim of image pre-processing is to enhance the image information containing unwanted distortions or to reinforce some image features for any processing. Preprocessing techniques use various techniques like dynamic image size and form, filtering of noise, image conversion, enhancing image and morphological operations.

✓ *Classification:*

Classification is employed for testing fruit diseases. The Random Forest classifier is utilized for classification.

- *Model Deployment using Flask: Website Creation with user Friendly Interface*

The trained CNN model was deployed using the Flask web framework in conjunction with HTML, CSS, and JavaScript to develop a user-friendly interface for disease detection. Flask's lightweight and scalable platform, coupled with its ability to seamlessly integrate with front-end technologies, makes it an ideal choice for hosting web applications that leverage machine learning models.

✓ *Flask Application Setup*

The Flask application was structured following the Model-View-Controller (MVC) pattern. It was configured with route definitions, static file directories, and template rendering settings to ensure smooth operation.

✓ *User Interface Design with Html and CSS*

HTML and CSS were utilized to design a visually appealing interface featuring upload buttons and image placeholders. CSS styling was applied for improved usability and aesthetics.

✓ *Enhancing Interactivity with Java Script*

JavaScript was used to enhance user interaction, implementing event listeners for image uploads and form submissions. Client-side validation mechanisms were employed to ensure a seamless user experience.

✓ *Integration with Flask Backend*

The Flask backend managed image preprocessing, model inference, and result rendering. Upon image upload, preprocessing steps such as resizing and normalization were applied before passing the image through the CNN model for diseaseidentification.

✓ *Real Time Feedback and Visualization*

The Flask application provided real-time feedback on disease presence in uploaded apple fruit images. Results from the model inference were visualized and displayed to users, enabling interaction and exploration of detected diseases.

✓ *Scalability and Deployment*

The Flask application was designed for scalability and deployment flexibility. Deployment options included local hosting for testing and development, as well as cloud-based solutions for broader accessibility and scalability across multiple platforms.

✓ *Real Time Interface*

"Real-time interface" refers to a user interface (UI) that provides immediate feedback or response to user input or changes in data. In the context of our CNN model for fruit disease detection, implementing a real-time interface would allow users, such as farmers or agricultural practitioners, to interact with the system seamlessly and receive timely feedback on disease detection results.

➢ *Tools and Technologies*

- *Python*

The Python community is large and friendly, and Stack Overflow is a great place to start looking for solutions to different problems you may run into when developing. It isone of the most widely used languages on the website, so there's a good chance you can find direct answers to any questions. Additionally, Python provides a wealth of powerful tools designed specifically for scientific computing. Rapid iteration and significantly less code are required to implement specific functionalities thanks to the easily accessible and well-documented packages such as NumPy, Pandas, and SciPy. Python's ability to tolerate code that closely resembles pseudo code is one of its noteworthy features. This is helpful because it makes the process of implementing and testing pseudo code described in academic papers simpler. But there are some disadvantages to Python. It can be annoying to deal with the dynamic nature of the language and the well-known "duck typing" issue within packages. Unexpected return types from package methods, like an object that looks like an array instead of an actual array, can be problematic. Furthermore, a lot of trial-and-error testing may be necessaryto determine method return types due to the standard Python documentation's lack of clarity, which makes learning new Python packages and libraries more difficult.

- *Numpy*

NumPy is a Python package that provides sophisticated and scientific mathematical functions inside the Python environment. Known as the scientific computing core library (Geetharamani & Pandian, 2019), it provides a powerful multidimensional array object and makes integration with C and C++ easier. Among other things, NumPy is very helpful for linear algebra tasks and has random number generation capabilities. NumPy's array type is a useful data structure that enhances Python by being specifically designed for numerical calculations. Furthermore, NumPy provides basic arithmetic operations, such as Eigenvector computation functions.

- *Scikit Learn*

A free machine learning library for Python called Scikit-learn (Mohanty et al. 2016) has many algorithms for tasks like classification, regression, and clustering. Among these algorithms are k-nearest neighbors, random forests, and support vector machines. Moreover, Scikit-learn interfaces with Python's scientific and numerical libraries, including SciPy and NumPy, with ease. Although Scikit-learn is mostly written in Python, Python is also used to implement some of the core algorithms for best performance.

- *Tensorflow*

TensorFlow is an open-source software library intended for numerical computation with data flow graphs (Geetharamani & Pandian, 2019). These graphs show nodes for mathematical operations and edges for the multidimensional data arrays (tensors) that are exchanged between them. Its adaptable architecture enables computation to be distributed across one or more CPUs or GPUs in desktop, server, or mobile devices using a single API. TensorFlow was first created for machine learning and deep neural network research by scientists and engineers in Google's Machine Intelligence research group, specifically the Google Brain Team. However, its adaptability makes it applicable to a wide range of fields outside of machine learning.

- *Keras*

A high-level neural network API that runs on top of the TensorFlow, CNTK, or Theanoframeworks, Keras (Ganatra, N. et al. 2020) is primarily designed in Python. It was designed with the intention of promoting quick experimentation, understanding that quick changes from idea to result are essential for carrying out meaningful research. Keras supports both convolutional and recurrent networks, as well as their combinations, to make prototyping simple and quick. It also runs computations on both CPU and GPU architectures with ease. A library comprising multiple implementations of commonly used neural network building blocks, such as layers, objectives, optimizers, and activation functions, is included in Keras, along with several tools to facilitate the handling of image and text data.

- *Compiler Option*

Anaconda is a top-tier open-source Python distribution designed for large-scale data processing, predictive analytics, and scientific computing applications. Simplifying the deployment and package management procedures is its main goal.

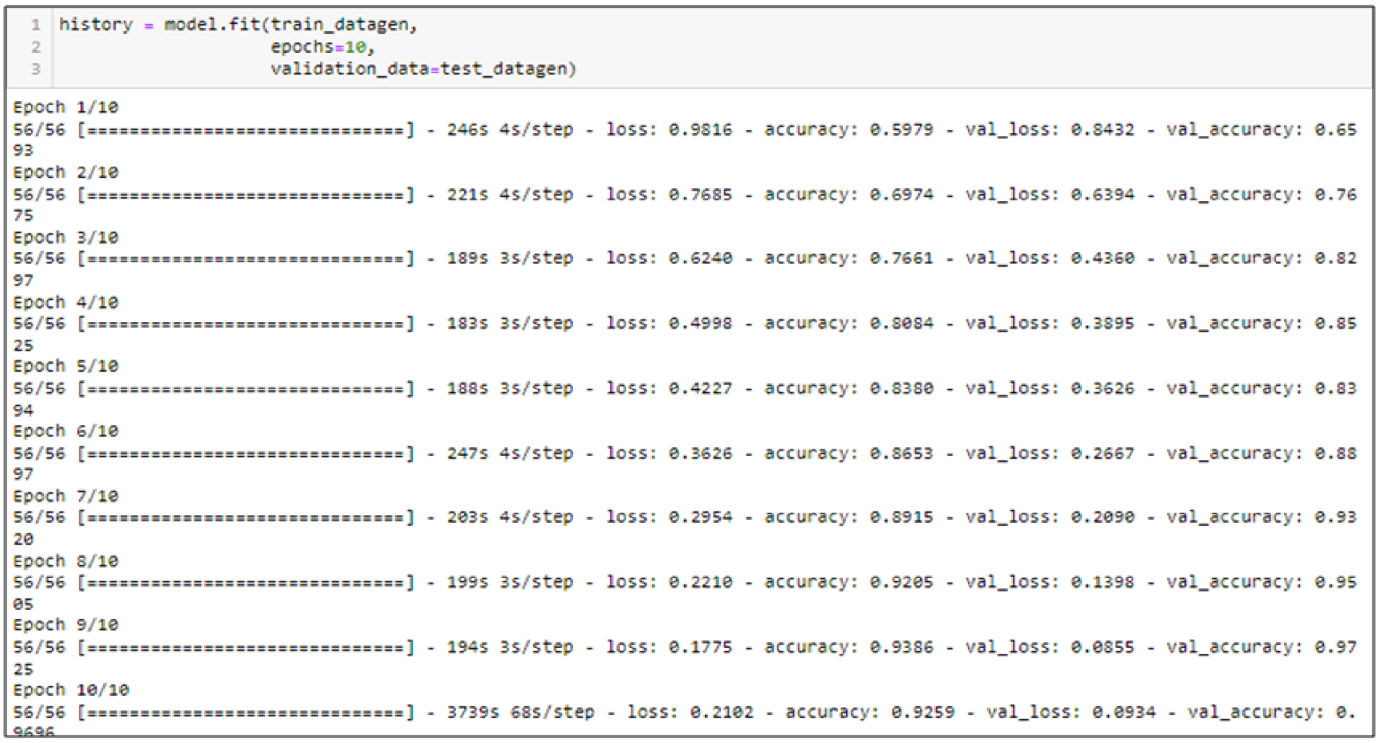
- *Jupyter Notebook*

The Jupyter Notebook is an open-source web application that makes it easier to create and share documents with interactive code, equations, graphics, and text. Its broad range of applications is applicable to a variety of fields, such as machine learning, statistical modeling, data visualization, numerical simulation, and data cleaning and transformation (Brownlee, J. 2016).

IV. RESULTS AND DISCUSSION

➢ *Results*

Achieving such a high accuracy rate holds significant implications, particularly in agriculture, where early disease detection can greatly influence crop yield and quality. A CNN model with a 95.37% accuracy rate shows promise for practical deployment in real-world scenarios, enabling timely interventions to mitigate disease spread and minimize yield losses. In this implementation, the proposed CNN model is evaluated for its accuracy. The training accuracy is plotted along with model validation accuracy v/s number of epochs in Fig. 9 It can be observed that there is a small difference in training and validation accuracy curves. It shows that the proposed model fits well for the addressed problem. It can also be observed from the graph that model accuracy initially increases sharply with increasing epochs. Later it improves slowly.



```
history = model.fit(train_datagen,
                    epochs=10,
                    validation_data=test_datagen)
```

```
Epoch 1/10
56/56 [==============================] - 246s 4s/step - loss: 0.9816 - accuracy: 0.5979 - val_loss: 0.8432 - val_accuracy: 0.65
93
Epoch 2/10
56/56 [==============================] - 221s 4s/step - loss: 0.7685 - accuracy: 0.6974 - val_loss: 0.6394 - val_accuracy: 0.76
75
Epoch 3/10
56/56 [==============================] - 189s 3s/step - loss: 0.6240 - accuracy: 0.7661 - val_loss: 0.4360 - val_accuracy: 0.82
97
Epoch 4/10
56/56 [==============================] - 183s 3s/step - loss: 0.4998 - accuracy: 0.8084 - val_loss: 0.3895 - val_accuracy: 0.85
25
Epoch 5/10
56/56 [==============================] - 188s 3s/step - loss: 0.4227 - accuracy: 0.8380 - val_loss: 0.3626 - val_accuracy: 0.83
94
Epoch 6/10
56/56 [==============================] - 247s 4s/step - loss: 0.3626 - accuracy: 0.8653 - val_loss: 0.2667 - val_accuracy: 0.88
97
Epoch 7/10
56/56 [==============================] - 203s 4s/step - loss: 0.2954 - accuracy: 0.8915 - val_loss: 0.2090 - val_accuracy: 0.93
20
Epoch 8/10
56/56 [==============================] - 199s 3s/step - loss: 0.2210 - accuracy: 0.9205 - val_loss: 0.1398 - val_accuracy: 0.95
05
Epoch 9/10
56/56 [==============================] - 194s 3s/step - loss: 0.1775 - accuracy: 0.9386 - val_loss: 0.0855 - val_accuracy: 0.97
25
Epoch 10/10
56/56 [==============================] - 3739s 68s/step - loss: 0.2102 - accuracy: 0.9259 - val_loss: 0.0934 - val_accuracy: 0.
9696
```

Fig 8 Model Training

- *Model Validation Accuracy*

Model validation accuracy represents the degree of correctness of predictions made by the model concerning the true labels or outcomes (Hasan et al., 2022). It is typically expressed as a percentage and reflects the proportion of correct predictions out of the total number of predictions made. Higher accuracy values indicate that the model is making more accurate predictions, thereby demonstrating its proficiency in understanding and capturing patterns within the data. However, it's crucial to note that accuracy alone may not provide a comprehensive understanding of the model's performance, especially in scenarios where class imbalances exist or when certain classes are more critical than others. Thus, while accuracy serves as a primary indicator of model performance, it should be complemented with other evaluation metrics for a more holistic assessment.

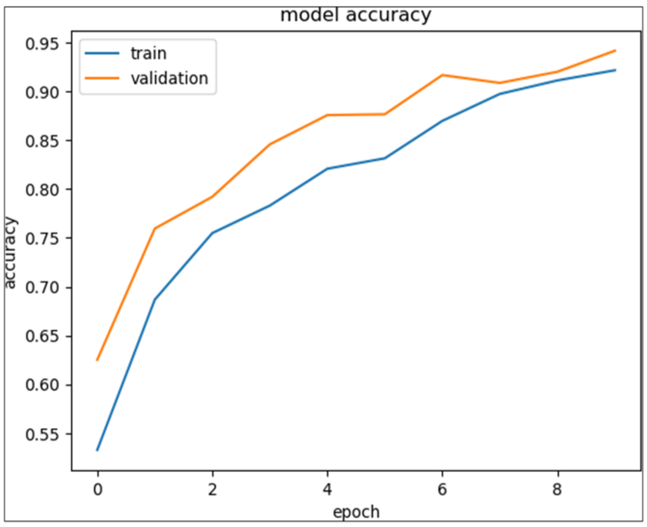


Fig 9 Model Validation Accuracy v/s No. of Epochs Curve

- *Model Loss*

Model loss (Hossein Azgomi et al. 2023), often referred to as the objective function or cost function, quantifies the disparity between the model's predictions and the actual ground truth labels. It represents the error incurred by the model during training and is minimized through optimization techniques such as gradient descent. Model loss is a crucial metric as it guides the learning process by signaling to the model how far off its predictions are from the true labels. Lower values of loss indicate that the model is converging towards better performance, effectively capturing the underlying patterns in the data. Different loss functions are employed based on the nature of the task, such as mean squared error (MSE) for regression tasks and categorical cross-entropy for classification tasks.

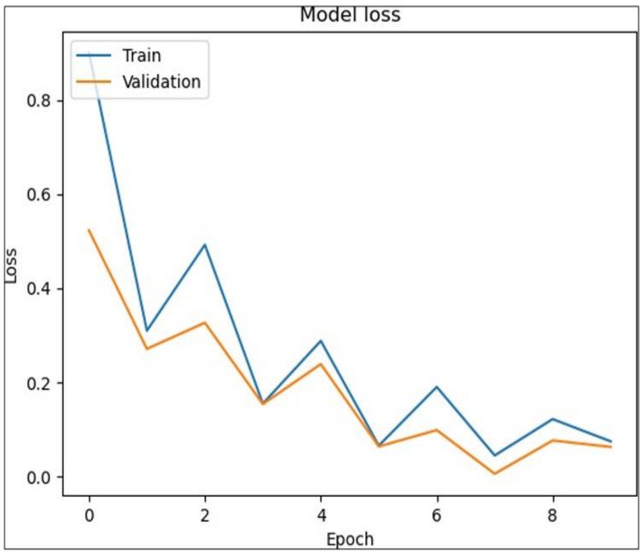


Fig 10 Model Loss v/s No. of Epochs Curve

- *Confusion Matrix*

A classification model's performance is graphically summarized by the confusion matrix (Peng Jiang et al. 2019), which shows accurate predictions on the diagonal and incorrect classifications off-diagonal. It helps identify difficult classes and offers insights into classification accuracy. The performance of the model can only be improved and maximized with the use of this tool. When presented with several classes that share a similar shape, classifiers may become confused. Reduced performance may also arise from highly complex patterns presented in the same class due to infected apple images at different stages or against different backgrounds. A model's classification accuracy can be visually estimated using a confusion matrix. The final test results' confusion matrix is displayed in Figure 11. The model's prediction accuracy in the relevant class increases with deeper coloration in the visualization results. Since every accurate prediction is on the diagonal and every inaccurate prediction is off the diagonal, it is easy to identify which classes have confused the detecting system.

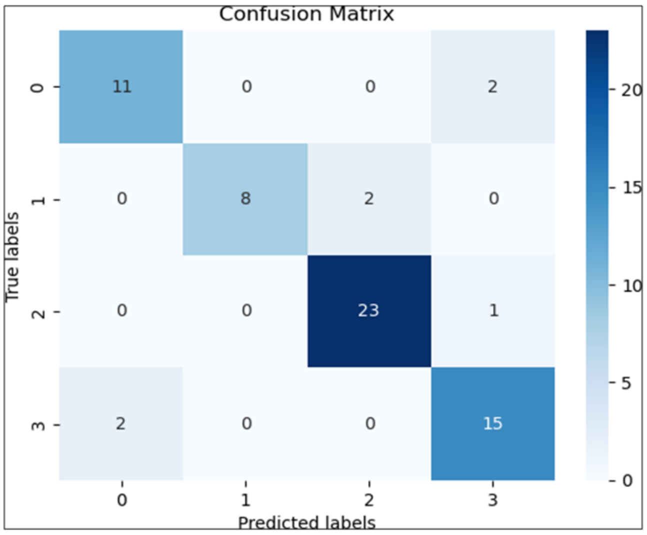


Fig 11 Confusion Matrix

- *Model Prediction Sample*

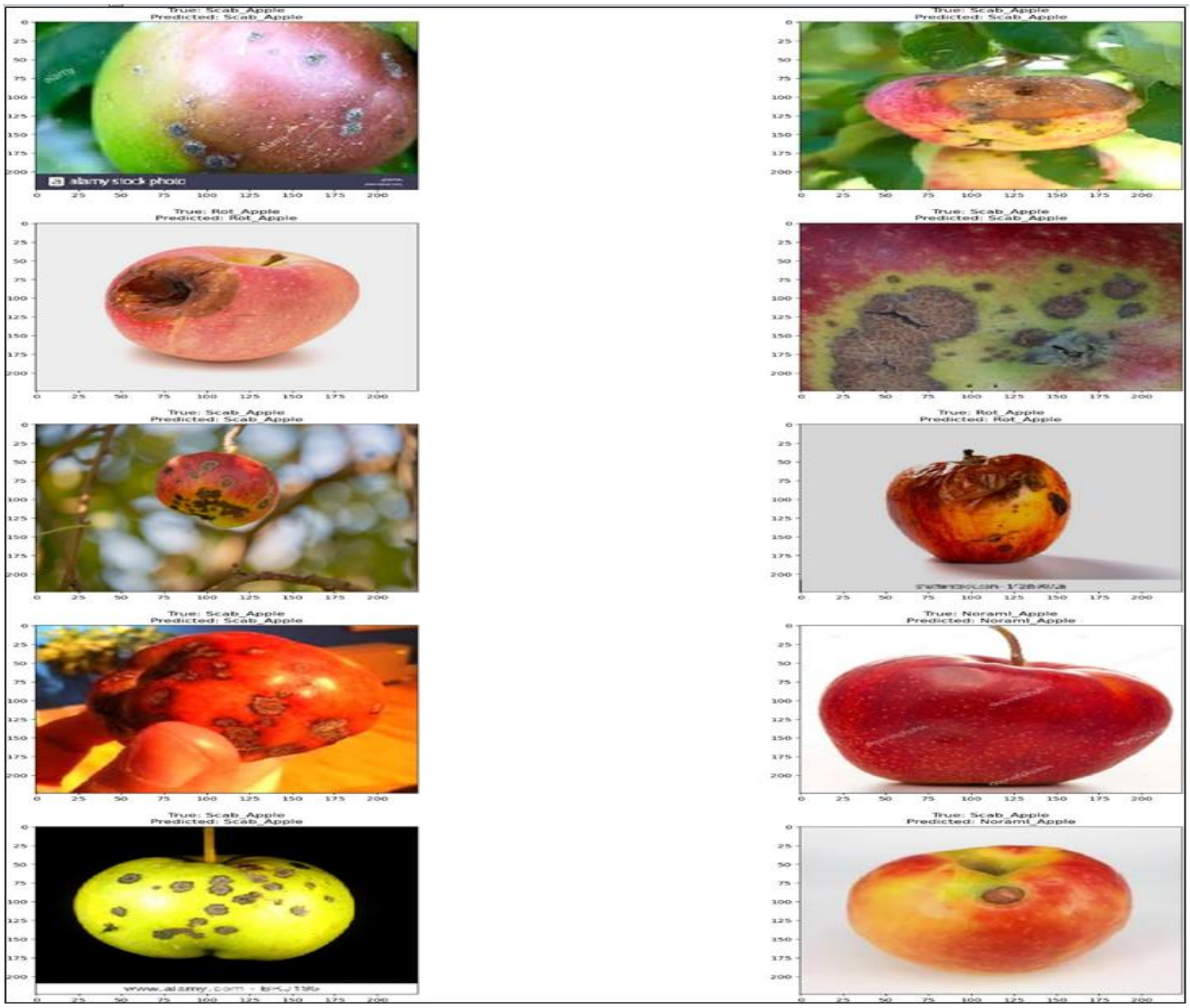


Fig 12 Model Prediction

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUL195

- *Model Deployment using Flask*

In the development of our Flask-powered website interface for the CNN image classification model, our main focus was on enhancing usability, functionality, and visual appeal. The homepage serves as the entry point, providing users with a brief project overview and a clear call-to-action (CTA) button leading them to the image classification tool. Using Flask, this tool enables users to upload images for classification, complete with straightforward instructions and visual indicators like progress bars and loading icons. I've ensured that classification results are transparently presented, showing predicted classes. Specifically, users can expect predictions for four classes of apple diseases: apple blotch, apple scab, apple rot, and normal apple. Additionally, I've integrated detailed documentation and help resources directly within the Flask application, aiming to assist users in understanding and effectively utilizing the classification tool. The design prioritizes a visually pleasing color scheme, typography, and responsive layout, enhancing accessibility and performance across various devices for a seamless user experience. Leveraging Flask's capabilities, the website interface operates in real-time, allowing users to interact with the classification tool immediately.

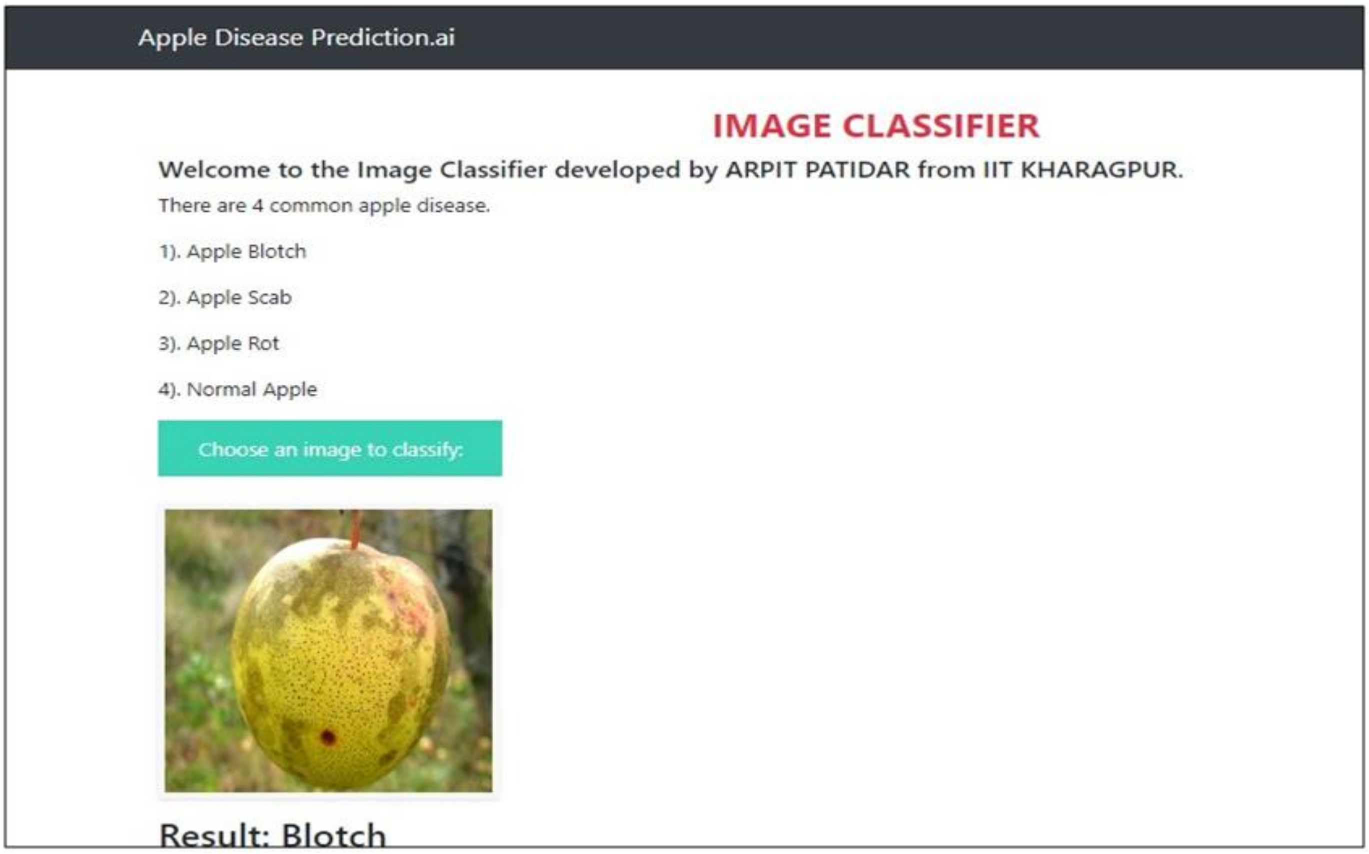

Apple Disease Prediction.ai

IMAGE CLASSIFIER

Welcome to the Image Classifier developed by ARPIT PATIDAR from IIT KHARAGPUR.

There are 4 common apple disease.

1). Apple Blotch

2). Apple Scab

3). Apple Rot

4). Normal Apple

Result: Blotch

Fig 13 Website Interface

➢ *Performance Metrics used for Model Evaluation*

The model performance is evaluated in respect of various metrics which can be derived from the confusion matrix.

- *Accuracy:*

Accuracy (0.9537) is a widely used performance metric in classification tasks. It measures the ratio of correct predictions to the total number of predictions made by the classifier. The confusion matrix enables the determination of both overall and individual class accuracies.

- *Precision:*

Precision (0.9537) measures the proportion of positive predictions that are correctly identified by the classifier. It evaluates the accuracy of positive predictions by calculating the ratio of true positives to the total number of predicted positives.

- *Recall or Sensitivity:*

Recall, or sensitivity (0.9531), assesses the classifier's capability to identify all positive samples accurately. It is determined by the ratio of true positive predictions to the total number of actual positive samples.

- *Specificity / True Negative Rate (TNR):*

Specificity, or True Negative Rate (TNR), measures the accuracy of a classifier in identifying negative samples. It is computed by dividing the number of correctly predicted negative samples by the total number of actual negative samples.

- *F1-Score:*

F1-Score (0.9525) gives the harmonic mean of the precision and recall, and refers to the no. of correctly classified instances. It ranges from 0 to 1.

V. CONCLUSION AND FUTURE WORK

➢ *Conclusions*

In our study, I delved into the feature-based approach utilized in existing systems, predominantly employing image processing techniques. I thoroughly examined steps such as image acquisition, preprocessing, feature extraction, and classification. To enhance disease detection in apples, I proposed a novel system utilizing Convolutional Neural Networks (CNNs). Through extensive experimentation, our CNN-based approach achieved a commendable accuracy of 95.37%. This result underscores the effectiveness of CNNs in disease identification within apple fruit, showcasing their potential for advancing agricultural disease management systems.

➢ *Future Work*

- Incorporation of More Fruit Varieties: Extend disease detection to various fruits, broadening agricultural applications.
- Mobile Application: Enable farmers to access our model easily through a mobile app.
- Cloud Integration: Utilize cloud services for enhanced scalability and collaboration, facilitating real-time updates and accessibility.

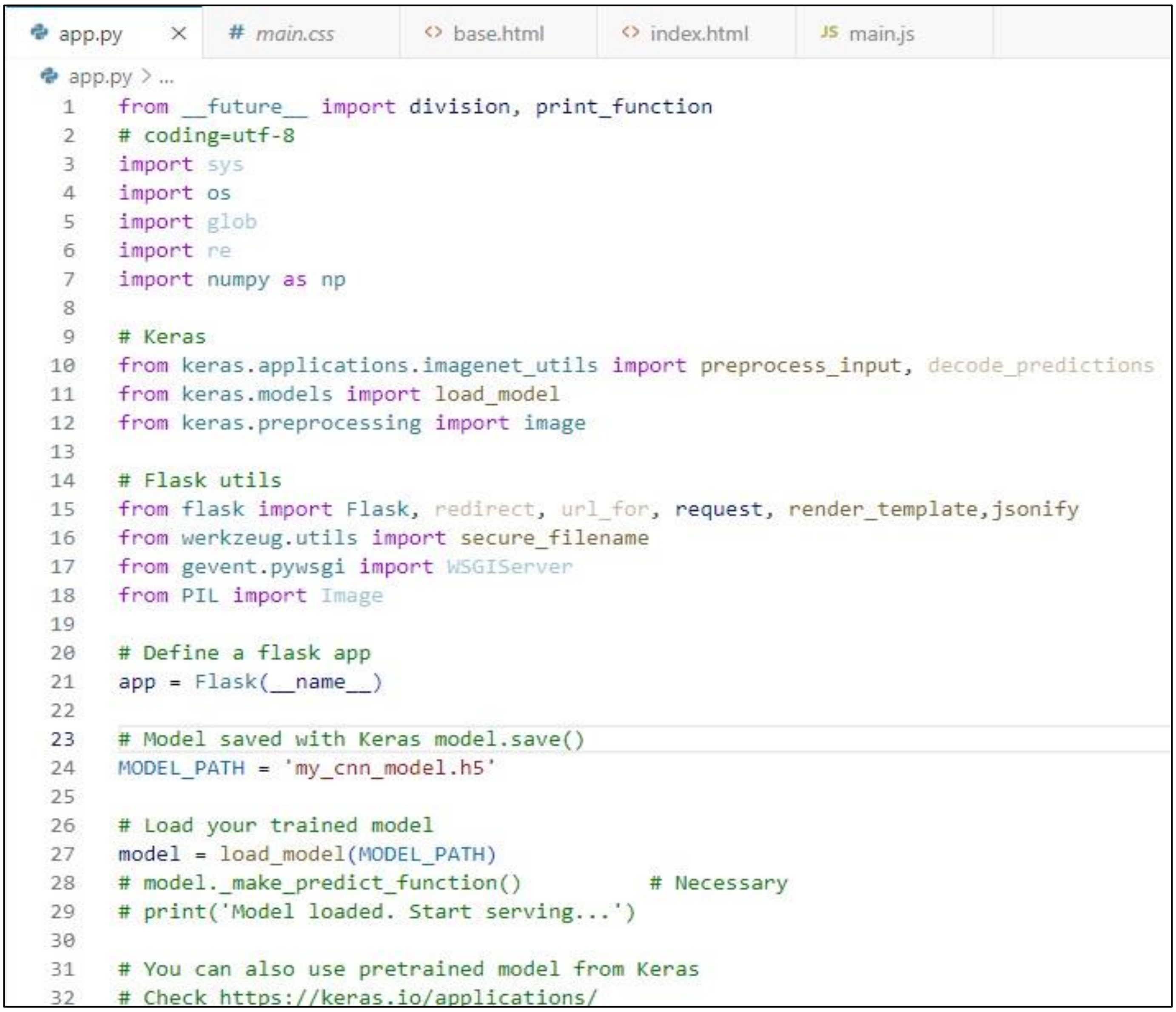
REFERENCES

[1]. Abdu, A. M., Mokji, M. M., Sheikh, U. U., & Khalil, K. (2019, March). Automatic disease symptoms segmentation optimized for dissimilarity feature extraction in digital photographs of plant leaves. In 2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA) (pp. 60-64). IEEE.
[2]. Adedoja, A., Owolawi, P. A., & Mapayi, T. (2019, August). Deep learning based on nasnet for plant disease recognition using leave images. In 2019 international conference on advances in big data, computing and data communication systems (icABCD) (pp. 1-5). IEEE.
[3]. Al Bashish, D., Braik, M., & Bani-Ahmad, S. (2011). Detection and classification of leaf diseases using K-means-based segmentation and. Information technology journal, 10(2), 267-275.
[4]. 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: CIGRJournal, 15(1), 211-217.
[5]. 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: CIGRJournal, 15(1), 211-217.
[6]. Awate, A., Deshmankar, D., Amrutkar, G., Bagul, U., & Sonavane, S. (2015, October). Fruit disease detection using color, texture analysis and ANN. In 2015 international conference on green computing and internet of things (ICGCIoT) (pp. 970-975). IEEE.
[7]. Balakrishna, K., & Rao, M. (2019). Tomato plant leaves disease classification using KNN and PNN. International Journal of Computer Vision and Image Processing (IJCVIP), 9(1), 51-63.
[8]. Brownlee, J. (2016). Introduction to the Python Deep Learning Library Theano. [Online]. Available.
[9]. De Luna, R. G., Dadios, E. P., & Bandala, A. A. (2018, October). Automated image capturing system for deep learning-based tomato plant leaf disease detection and recognition. In TENCON 2018-2018 IEEE Region 10 Conference (pp. 1414-1419). IEEE.
[10]. Devaraj, A., Rathan, K., Jaahnavi, S., & Indira, K. (2019, April). Identification of plant disease using image processing technique. In 2019 International Conferenceon Communication and Signal Processing (ICCSP) (pp. 0749-0753). IEEE.
[11]. Dutot, M., Nelson, L. M., & Tyson, R. C. (2013). Predicting the spread of postharvest disease in stored fruit, with application to apples. Postharvest biology and technology, 85, 45-56.
[12]. Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and electronics in agriculture, 145, 311-318.
[13]. Ganatra, N., & Patel, A. (2020). A multiclass plant leaf disease detection using image processing and machine learning techniques. International Journal on Emerging Technologies, 11(2), 1082-1086.
[14]. Geetharamani, G., & Pandian, A. (2019). Identification of plant leaf diseases using a nine-layer deep convolutional neural network. Computers & Electrical Engineering, 76, 323-338.
[15]. Han, J., & Moraga, C. (1995, June). The influence of the sigmoid function parameters on the speed of backpropagation learning. In International workshop on artificial neural networks (pp. 195-201). Berlin, Heidelberg: Springer BerlinHeidelberg.
[16]. Ilic, M., Spalevic, P., Veinovic, M., & Ennaas, A. A. M. (2015, November). Data mining model for early fruit diseases detection. In 2015 23rd Telecommunications Forum Telfor (TELFOR) (pp. 910-913). IEEE.
[17]. Jhuria, M., Kumar, A., & Borse, R. (2013, December). Image processing for smart farming: Detection of disease and fruit grading. In 2013 IEEE secondinternational conference on image information processing (ICIIP-2013) (pp.521-526). IEEE.
[18]. Jiang, P., Chen, Y., Liu, B., He, D., & Liang, C. (2019). Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. IEEE Access, 7, 59069-59080.
[19]. Kim, M. S., Lefcourt, A. M., Chen, Y. R., & Tao, Y. (2005). Automated detection of fecal contamination of apples based on multispectral fluorescence image fusion. Journal of food engineering, 71(1), 85-91.

[20]. Kleynen, O., Leemans, V., & Destain, M. F. (2005). Development of a multi-spectral vision system for the detection of defects on apples. Journal of food engineering, 69(1), 41-49.
[21]. Kulkarni, O. (2018, August). Crop disease detection using deep learning. In 2018 Fourth international conference on computing communication control and automation (ICCUBEA) (pp. 1-4). IEEE.
[22]. Kumar, A., & Gill, G. S. (2015, May). Automatic fruit grading and classification system using computer vision: A review. In 2015 Second International Conference on Advances in Computing and Communication Engineering (pp. 598-603). IEEE.
[23]. Lee, S. H., Wu, C. C., & Chen, S. F. (2018). Development of image recognition and classification algorithm for tea leaf diseases using convolutional neuralnetwork. In 2018 ASABE Annual International Meeting (p. 1). American Society of Agricultural and Biological Engineers.
[24]. Liu, B., Zhang, Y., He, D., & Li, Y. (2017). Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry, 10(1), 11.
[25]. Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2017). Identification of rice diseases using deep convolutional neural networks. Neurocomputing, 267, 378-384.
[26]. Mahlein, A. K., Rumpf, T., Welke, P., Dehne, H. W., Plümer, L., Steiner, U., & Oerke,
[27]. E. C. (2013). Development of spectral indices for detecting and identifying plant diseases. Remote Sensing of Environment, 128, 21-30.
[28]. Mehl, P. M., Chao, K., Kim, M., & Chen, Y. R. (2002). Detection of defects on selected apple cultivars using hyperspectral and multispectral image analysis. Applied engineering in agriculture, 18(2), 219.
[29]. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 215232.
[30]. Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on pattern analysis and machine intelligence, 24(7), 971-987.
[31]. Patel, S., Jaliya, U. K., & Patel, P. (2020). A survey on plant leaf disease detection. Int.J. Mod. Trends Sci. Technol, 6, 129-134.
[32]. Qin, F., Liu, D., Sun, B., Ruan, L., Ma, Z., & Wang, H. (2016). Identification of alfalfa leaf diseases using image recognition technology. PloS one, 11(12), e0168274.
[33]. Sahithya, V., Saivihari, B., Vamsi, V. K., Reddy, P. S., & Balamurugan, K. (2019, April). GUI based detection of unhealthy leaves using image processing techniques. In 2019 International Conference on Communication and Signal Processing (ICCSP) (pp. 0818-0822). IEEE.
[34]. Soni, P., & Chahar, R. (2016, July). A segmentation improved robust PNN model for disease identification in different leaf images. In 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES) (pp. 1-5). IEEE.
[35]. Suma, V., Shetty, R. A., Tated, R. F., Rohan, S., & Pujar, T. S. (2019, June). CNN based leaf disease identification and remedy recommendation system. In 2019 3rd International conference on Electronics, Communication and AerospaceTechnology (ICECA) (pp. 395-399). IEEE.
[36]. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
[37]. Tm, P., Pranathi, A., SaiAshritha, K., Chittaragi, N. B., & Koolagudi, S. G. (2018, August). Tomato leaf disease detection using convolutional neural networks. In 2018 eleventh international conference on contemporary computing (IC3) (pp. 1-5). IEEE.
[38]. Waghmare, H., Kokare, R., & Dandawate, Y. (2016, February). Detection and classification of diseases of grape plant using opposite colour local binary pattern feature and machine learning for automated decision support system. In 2016 3rd international conference on signal processing and integrated networks (SPIN) (pp. 513-518). IEEE.
[39]. Wu, C., Luo, C., Xiong, N., Zhang, W., & Kim, T. H. (2018). A greedy deep learning method for medical disease analysis. IEEE Access, 6, 20021-20030.

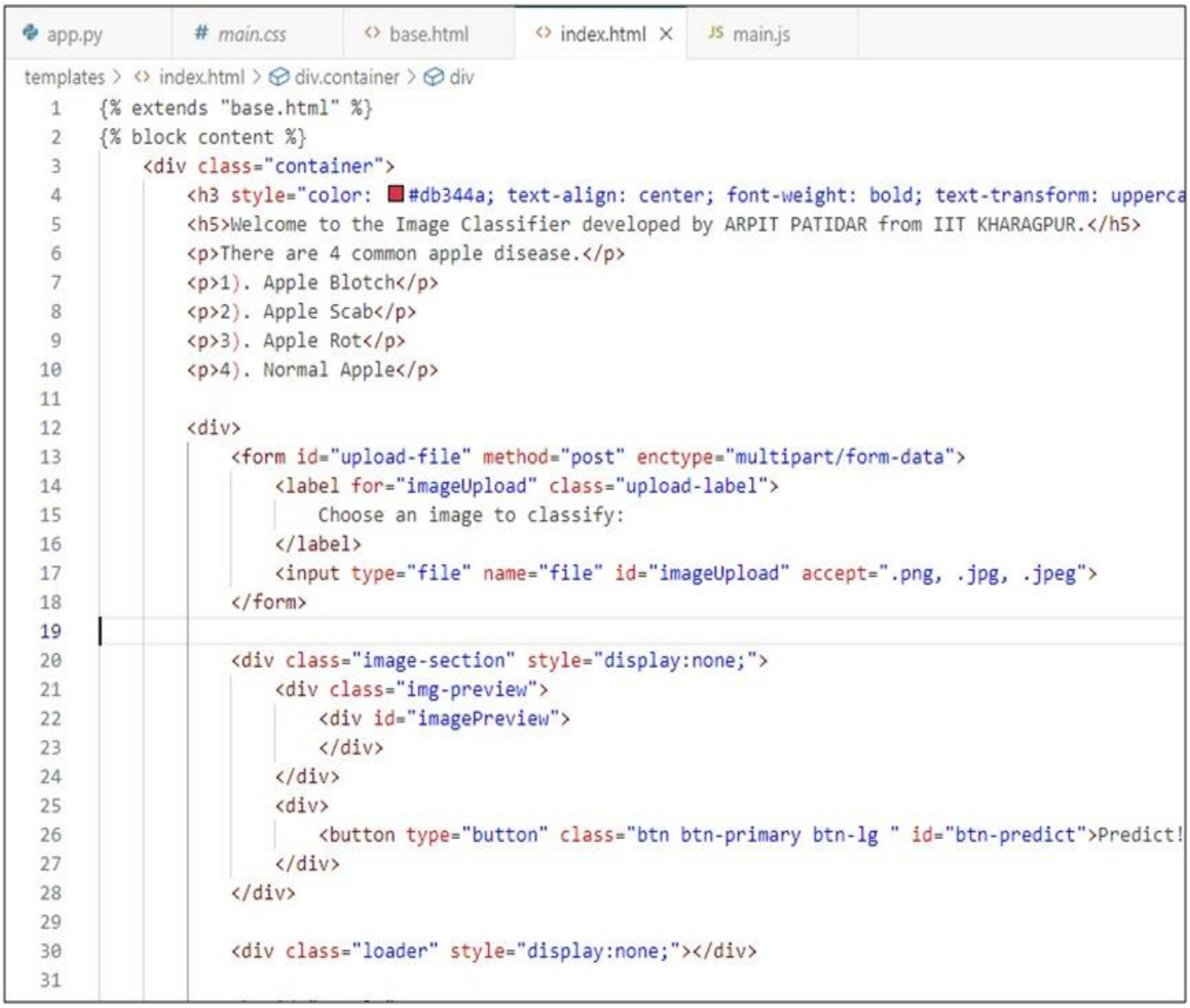
APPENDIX

FLASK APP PROGRAMMING CODE



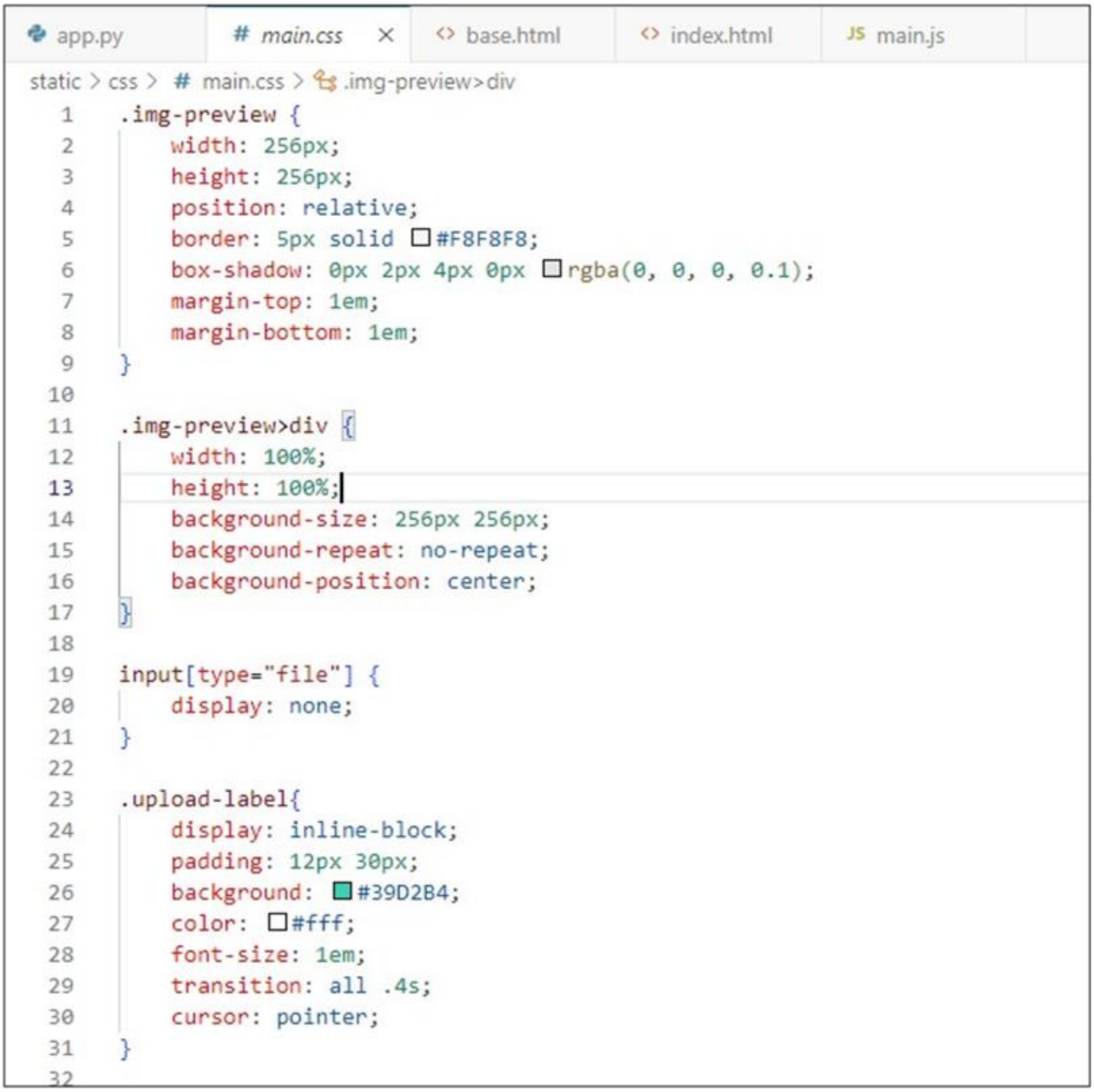
```python
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import re
import numpy as np

# Keras
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras.preprocessing import image

# Flask utils
from flask import Flask, redirect, url_for, request, render_template,jsonify
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
from PIL import Image

# Define a flask app
app = Flask(__name__)

# Model saved with Keras model.save()
MODEL_PATH = 'my_cnn_model.h5'

# Load your trained model
model = load_model(MODEL_PATH)
# model._make_predict_function()          # Necessary
# print('Model loaded. Start serving...')

# You can also use pretrained model from Keras
# Check https://keras.io/applications/
```

HTML PROGRAMMING CODE



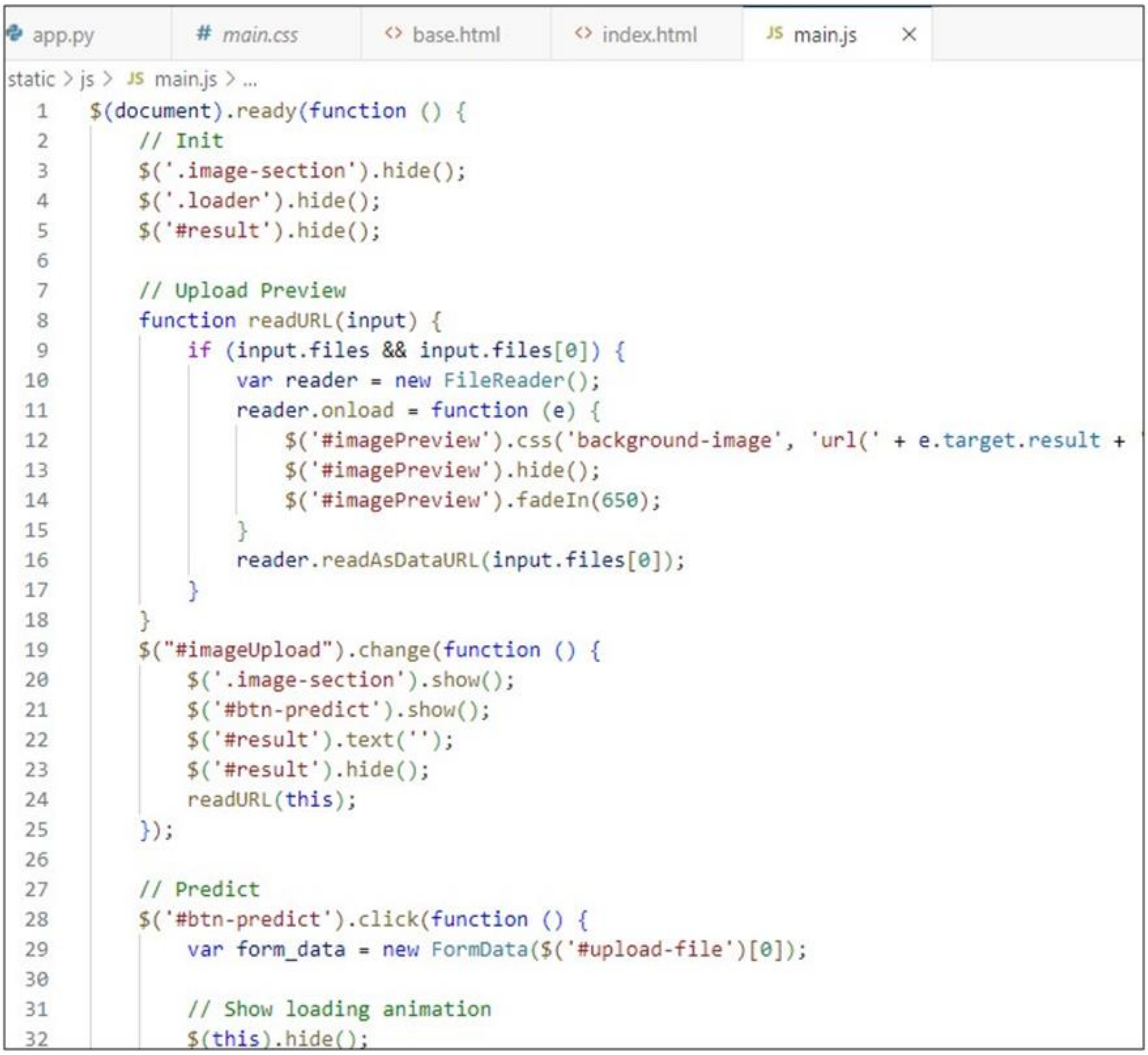
```
{% extends "base.html" %}
{% block content %}
    <div class="container">
        <h3 style="color: #db344a; text-align: center; font-weight: bold; text-transform: upperca
        <h5>Welcome to the Image Classifier developed by ARPIT PATIDAR from IIT KHARAGPUR.</h5>
        <p>There are 4 common apple disease.</p>
        <p>1). Apple Blotch</p>
        <p>2). Apple Scab</p>
        <p>3). Apple Rot</p>
        <p>4). Normal Apple</p>

        <div>
            <form id="upload-file" method="post" enctype="multipart/form-data">
                <label for="imageUpload" class="upload-label">
                    Choose an image to classify:
                </label>
                <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
            </form>

            <div class="image-section" style="display:none;">
                <div class="img-preview">
                    <div id="imagePreview">
                    </div>
                </div>
                <div>
                    <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!
                </div>
            </div>

            <div class="loader" style="display:none;"></div>

```

CSS PROGRAMMING CODE



```css
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}

```

JAVA SCRIPT PROGRAMMING CODE



```
$(document).ready(function () {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result +
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            }
            reader.readAsDataURL(input.files[0]);
        }
    }
    $("#imageUpload").change(function () {
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });

    // Predict
    $('#btn-predict').click(function () {
        var form_data = new FormData($('#upload-file')[0]);

        // Show loading animation
        $(this).hide();
```