ESP Journal of Engineering & Technology Advancements (ESP-JETA)
ISSN : 2583-2646
Special Issue

ENUMERATION OF RICE PLANT DISEASE DETECTION BY USING MOBILE APP

Karthigailakshmi B[1], Nivetha S[2], Sutharshini S[3,] Niraanjanadevi Jeevanandham[4]
Department of Agriculture Engineering, Mangaiyarkarasi college of Engineering, Paravai, Madurai
Email:sutharshini2002@gmail.com

Abstract*:* Rice plant disease detection mobile apps utilize cutting-edge technologies such as image processing, machine learning, and artificial intelligence to enable farmers to diagnose diseases accurately and swiftly. These apps typically feature user-friendly interfaces that allow farmers to capture images of diseased plants using their smartphone cameras. Once an image is captured, the app employs advanced algorithms to analyze the visual characteristics of the plant, such as leaf discoloration, lesions, and deformities, to determine the presence of any diseases. By comparing the captured images with extensive databases of plant diseases and symptoms, the app can provide real-time diagnosis and recommend appropriate treatment measures to mitigate the spread of the disease and minimize crop losses.

Keywords: Mobile App Multi disease detection paddy (BB, Blast, sheath blight, false smut, tungro disease were identified as real time method by using **PAUDITECT** app)

INTRODUCTION

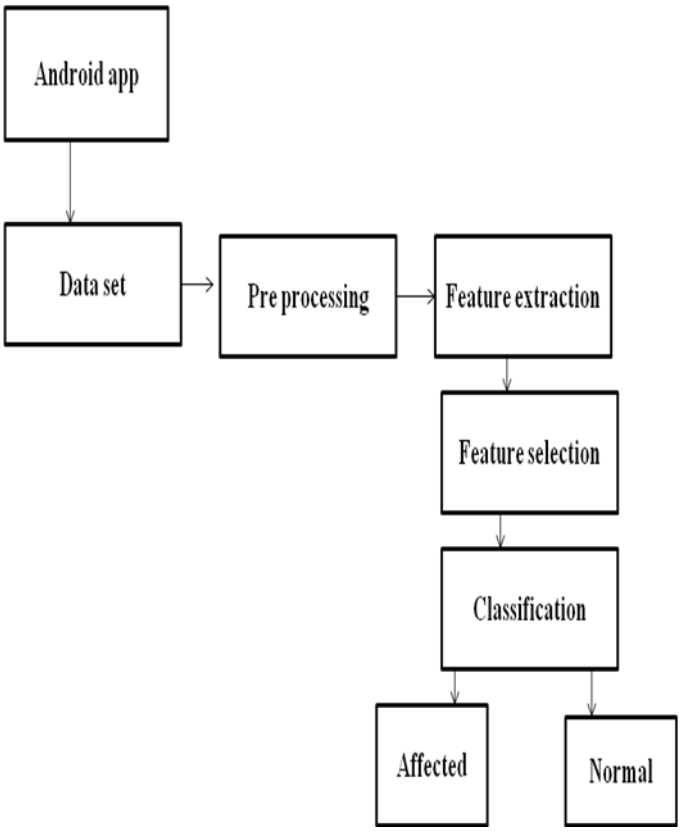
Rice plant disease detection using mobile apps represents a promising approach to enhancing disease management practices in agriculture. By leveraging the capabilities of modern technology, these apps offer farmers a convenient, accessible, and effective tool to diagnose and manage plant diseases, thereby improving crop health, reducing crop losses, andcontributing to global food securityefforts. Advancements in mobile technology continue to evolve, plant disease detection apps hold immense potential to revolutionize agricultural practices and empower farmers to meet the challenges of feeding a growing population in a

sustainable manner. The paddy is the main crop we have selected in our project. Rice Plant disease detection mobile apps also contribute to sustainable agriculture by promoting precision disease management strategies

LITERATURE SURVEY

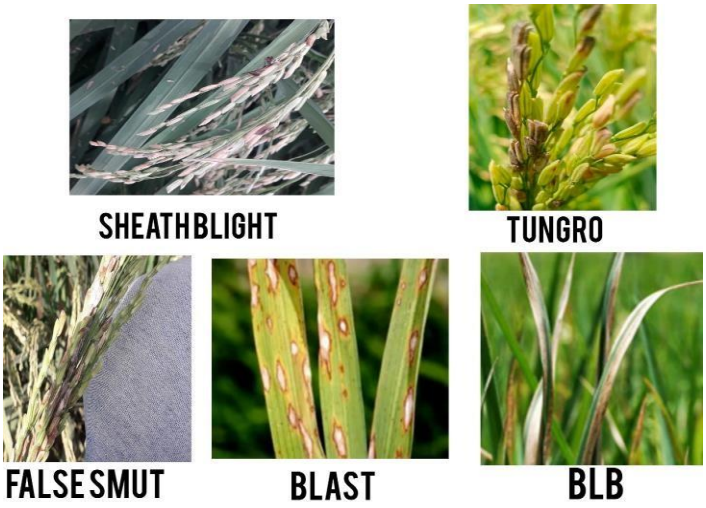
"Deep Learning-Based Plant Disease Detection Using Mobile Applications" on the Smith,(J., et al. 2020): This study utilized deep learning techniques to develop a mobile application for plant disease detection. The researchers trained convolution neural networks (CNNs) on large datasets of diseased and healthy plant images to enable accurate disease diagnosis."Image Processing-Based Plant Disease Detection on Mobile Platforms" Patel,(A., et al 2019).This research focused on utilizing image processing techniques for plant disease detection on mobile platforms. "Machine Learning Approaches for Plant Disease Diagnosis via Mobile Applications. Kumar,(R., et al.2021).This study explored the application of machine learning approaches for plant disease diagnosis through mobile applications.: "Fusion of Image Processing and Machine Learning for Plant Disease Detection on Mobile Devices "Gupta,(S., et al.at 2018).This research proposed a fusion approach combining image processing and machine learning techniques for plant disease detection on mobile devices."Internet of Things (IOT)-Enabled plant Disease Detection on Mobile Platforms"Wang,(Y., et al.at 2019). This research investigated the integration of Internet of Things (IOT) technology with mobile platforms for plant disease detection.

PROPOSED SYSTEM



DATA SET COLLECTION

The dataset is a collection of labelled images of both healthy and diseased plants used to train and evaluate the disease detection model. It consists of images depicting various types of plant diseases, along with corresponding metadata indicating the type of disease present. The dataset serves as the foundation for building a robust and accurate disease detection system by providing the necessary input for training machine learning algorithms.



PRE-PROCESSING

Pre-processing refers to the initial stage of data preparation, where raw image data is transformed and cleaned to improve the performance of the disease detection model. This may involve tasks such as resizing images to a consistent resolution, normalizing pixel values, and removing noise or artifacts from the images. Preprocessing helps ensure that the input data is standardized and suitable for further analysis and feature extraction.

FEATURE EXTRACTION

Feature extraction involves extracting meaningful information or descriptors from the pre-processed image data that can be used as input features for the disease detection model. This may include techniques such as histogram of oriented gradients (HOG), local binary patterns (LBP), or color histograms. Feature extraction aims to capture relevant patterns and structures in the image data that are indicative of different plant diseases, facilitating accurate classification by the machine learning model.

CLASSIFICATION:

Classification is the final stage of the disease detection process, where the extracted features are used to classify the input images into different disease categories. Machine learning algorithms, such as support vector machines (SVM), decision trees, or convolutional neural networks (CNNs), are trained on the labelled dataset to learn patterns and relationships between input features and disease classes. Classification enables the Android app to provide real-time diagnosis and recommendations to farmers based on the detected plant diseases, empowering them to take timely actions to protect their crops.

PROGRAMME:

```
fromgoogle.colab import drive
drive.mount('/content/gdrive')
importos
os.chdir("/content/gdrive/MyDrive")
importzipfile
importos
importshutil
withzipfile.ZipFile("paddy_Database.zip","r") as f:
f.extractall('.')
importtensorflow as tf
fromtensorflow import keras
importmatplotlib.pyplot as plt
importnumpy as np
importos
base_dir = "/content/gdrive/MyDrive/Paddy_Database"
image_size = 224
#Creating DataGenerator
train_datagen = keras.preprocessing.image.ImageDataGenerator(rescale = 1/255.0,
shear_range = 0.2,
zoom_range = 0.2,
width_shift_range = 0.2,
height_shift_range = 0.2,
fill_mode="nearest")
batch_size = 32
train_data = train_datagen.flow_from_directoryos.path.join(b
```

```
ase_dir,"train"),
target_size=(image_size,image_size),
batch_size=batch_size,
class_mode="categorical"
                                )
test_datagen                                    =
keras.preprocessing.image.ImageDataGenera
tor(r escale = 1/255.0)
test_data                                       =
test_datagen.flow_from_directory(os.path.j
oin(ba se_dir,"validation"),
target_size=(image_size,image_size),
batch_size=batch_size,
class_mode="categorical"
                                ) #GAN-
Generate # importing required modules
fromkeras.preprocessing.image
                                                i
mport ImageDataGenerator
fromkeras.models import Sequential
fromsklearn.metrics import classification_report,
confusion_matrix
fromkeras.models import load_model
fromkeras.callbacks import EarlyStopping
fromkeras.layers    import    Activation,
                    Dropout,
Flatten, Dense, Conv2D, MaxPooling2D
import warnings
fromkeras.callbacks import
ModelCheckpoint importtensorflow as tf
# CNN Architecture
model = tf.keras.models.Sequential([
   # Note the input shape is the desired size
of the image 150x150 with 3 bytes colour
   # this is the first convolution
tf.keras.layers.Conv2D(64, (3,3), padding='same',
activation='relu', input_shape=(224, 224, 3)),
tf.keras.layers.Conv2D(64, (3,3),
padding='same', activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
   # The second convolution
tf.keras.layers.Conv2D(128,

(3,3), padding='same', activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
   # The third convolution
tf.keras.layers.Conv2D(256,                    (3,3),
padding='same', activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
   # the fourth convolution
```

```
tf.keras.layers.Conv2D(512, (3,3), padding='same', activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
    # The Fifth convolution
tf.keras.layers.Conv2D(512, (3,3), padding='same', activation='relu'),
tf.keras.layers.MaxPooling2D(2, 2),
    # Flatten the results to feed into a DNN
tf.keras.layers.Flatten(),
#model = tf.keras.models.load_model('/content/gdrive/My Drive/Leaf_Database/leaf-cnn.h5')
fromkeras.preprocessing.image import ImageDataGenerator
fromkeras.models import Sequential
fromsklearn.metrics import classification_report, confusion_matrix
fromkeras.models import load_model
fromkeras.callbacks import EarlyStopping
fromkeras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D
import warnings
fromkeras.callbacks import ModelCheckpoint
importtensorflow as tf
importmatplotlib.pyplot as plt
importnumpy as np
import cv2
model = tf.keras.models.load_model('/content/gdrive/My Drive/leaf-cnn.h5')
image_path = "/content/gdrive/MyDrive/Leaf_Database/test/paddy.11.jpg"
imageObj = cv2.imread(image_path)
new_img = tf.keras.preprocessing.image.load_img(image_path, target_size=(224,224))
img = tf.keras.preprocessing.image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
plt.imshow(new_img)
plt.axis("off")
plt.title("Input Image")
plt.show()
# Get RGB data from image
blue_color = cv2.calcHist([imageObj], [0], None,
```

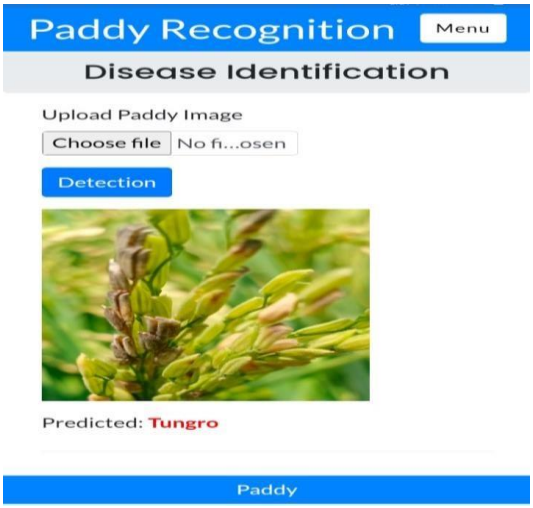
```
[256], [0, 256])
red_color = cv2.calcHist([imageObj], [1],
None, [256], [0, 256])
green_color = cv2.calcHist([imageObj], [2], None, [256], [0, 256])
plt.title("Histogram of all RGB
Colors") plt.hist(blue_color,
color="blue") plt.hist(green_color,
color="green") plt.hist(red_color,
color="red") plt.show()
img = img/255.0
prediction =
model.predict(img)
plt.axis("off")
plt.imshow(new_img)
plt.title(categories[np.argmax(prediction)])
```

RESULT AND DISCUSSION

An Android app is a software application designed to run on Android-powered devices, such as smart phones and tablets. In the context of plant disease detection, the Android app serves as the user interface through which farmers can interact with the disease detection system. It allows users to capture images of diseased plants using the device's camera. Then process the images using built-in algorithms, and receive real-time feedback on the presence of plant diseases. This feedback is helpful for farmers to enhance the quality of plant and also increase the production rate.

Opening page of app

Predict the plant disease by analyzing the image



Shows the information about the disease

CONCLUSION

Plant disease detection through mobile apps holds immense potential to empower farmers by providing them with accessible and timely information about the health of their crops. The deployment of mobile apps for plant disease detection contributes significantly to global food security efforts by enabling early detection and management of crop diseases. Mobile-based disease detection encourages the adoption of sustainable agricultural practices by reducing reliance on chemical pesticides and

promoting targeted interventions. The success of plant disease detection through mobile apps relies on collaboration among farmers, researchers, and developers to continuously improve the accuracy, effectiveness, and accessibility of these tools. By fostering a community-driven approach to agricultural innovation, these apps create opportunities for knowledge sharing, capacity building, and collective action towards a more resilient and sustainable agricultural future.

REFERENCES

- Gnanaprakasam, G., &Tamilselvan, M. (2020). "Plant Disease Detection and Classification Using Mobile Application." In 2020 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN) (pp. 1-5). IEEE.
- Mohanty, S. P., Hughes, D. P., &Salathé, M. (2016). "Using Deep Learning for Image-Based Plant Disease Detection." Frontiers in Plant Science, 7, 1419.
- Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). "Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification." Computational Intelligence and Neuroscience, 2016, 3289801.
- Singh, A., Ganapathysubramanian, B., Singh, A. K., & Sarkar, S. (2016). "Machine Learning for High-Throughput Stress Phenotyping in Plants." Trends in Plant Science, 21(2), 110-124.