ISSN:2229-6107

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY

E-mail :
editor.ijpast@gmail.com
editor@ijpast.in

www.ijpast.in

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 14,Issuse 2.May 2024

DETECTION OF TOMATO LEAF DISEASES FOR AGRO-BASED INDUSTRIES USING NOVEL PCA DEEPNET

NAKKALA MOUNIKA, PITTU BHUVANESHWARA REDDY

Abstract: The project employs advanced Deep Learning and Computer Vision techniques to improve the detection of diseases in tomato plants, crucial for optimizing agricultural production. Introducing the PCA DeepNet framework, the project combines classical Machine Learning with a custom Deep Neural Network, offering a unique and powerful methodology for capturing both data intricacies and complex disease patterns. Leveraging the Faster Region-Based Convolutional Neural Network (F-RCNN), the project achieves accurate classification of healthy and diseased tomato leaves, showcasing impressive classification accuracy and a robust mean average precision. Results demonstrate that the PCA DeepNet framework outperforms existing methods, promising a significant contribution to agro-based industries by providing highly accurate and reliable detection of tomato leaf diseases.This project extends its functionality with advanced models, including DenseNet, Xception, and a Voting Classifier for classification, alongside YoloV5 and YoloV8 for detection. The goal is to achieve exceptional accuracy for classification and high precision (mAP of 0.85 or above) for detection.

Index terms *-Tomato leaf diseases, artificial intelligence, deep learning, computer vision, generative adversarial networks, convolutional neural network, faster region-based convolutional neural network.*

INTRODUCTION

Agriculture is one of the oldest occupations practiced worldwide in the majority of the countries. It forms an important aspect of sustainability and survivability. India being a country primarily having an agriculture-based economy, hence a major part of India's population is either directly or indirectly linked to agriculture. Moreover, agricultural exports majorly contribute to the country's GDP. According to the available data, the global population is expected to reach 10 billion by2050 which in turn would require agricultural productivity to increase by at least 70%. With increase in world population, the demand for agricultural products has increased manifold.

Assistant professor[1], Dept of CSE, Chirala Engineering College, Chirala,
nakkalamounika2121@gmail.com
PG Student[2]-MCA, Dept of MCA, Chirala Engineering College, Chirala,
pittubhanu13@gmail.com

As further cultivable land area cannot be increased, hence the only way to increase the amount of agricultural production is to enhance the productivity of the existing lands. Agriculture in many parts of India is still practiced manually in a traditional manner, involving lots of manpower and manhours. Among several aspects of agriculture, keeping crops disease-free is an important facet. This is done manually and may turn out to be inaccurate owing to some limitations and incorrect judgments of humans which in this case might resultin catastrophic consequences; the worst being the whole crop getting ruined. This is a problem that no farmer can afford and hence resort to Artificial Intelligence (AI) based solutions. AI in agriculture serves a better pathway to analyze real-time problems faced by the farmers in day-to-day life. One of the common problems faced is the invasion of pests, which deteriorates the quality of the crops. The main challenge lies in detecting the diseases caused due to the attack of these pests and for which farmers need innovative technologies to combat such attacks. The joint venture of Computer Vision and Artificial Intelligence makes it a great way to solve such kinds of problems. The most promising factor that rules AI is that it uses real-time data not only to predict the emergence but also the identification of the pest and diseases before it takes a huge shape. Therefore, the main motto of developing such automated systems using AI is to reduce the vulnerability of pest attack and to preserve the quality of production.

Plant disease detection using Deep Learning techniques such as classification and detection has become a crucial aspect in monitoring and analyzing the productivity of each and every specific species of plant. As compared to traditional classification networks, techniques involving deep learning yields better results for real-time identification of plant leaf diseases [1]. These are all headed under the latest improvements in computer vision aided systems to efficiently provide solutions for multiple plant diseases as the existing method for disease detection is through naked eyes which require lots of effort and is time consuming as well [2]. Therefore to reduce this problem Deep Learning has been introduced which involves a robust process with higher accuracy for accurate diagnosis of the respective diseases [3]. The proposed work for detection of Tomato leaf diseases is accomplished with a Deep Learning approach combined with Machine Learning technique. The overall framework depicts the classification of Tomato leaf images into healthy and diseased ones and then implementation of the images for different categories of disease detection. The entire work has been implemented using Deep Neural Networks, especially using Convolution Neural Network (CNN) architectures and PCA and this new model is named as PCA DeepNet. The PCA work as the primary feature extractor followed by the customized deep neural networks for classification and detection purposes. The convolutional deep learning networks are basically chosen to reduce computational cost and for smooth classification; thereby helping in development of an intelligent systems assisted tomato leaf disease detection. A single shot detector (SSD) and Faster Region Based Convolutional Neural Network (F-RCNN) is used for detection purposes. In F-RCNN, the detection steps are carried out in two steps unlike SSD and hence a more accurate detection is obtained using F-RCNN [4].

1. LITERATURE SURVEY

ISSN 2229-6107 www.ijpast.in
Vol 14,Issuse 2.May 2024

The difference of leaf disease images within the class is large but the difference between the class is small, so it is very important to represent the features of the local area of the target [1. 4. 5. 6. 7]. Moreover, the complex network occupies a large amount of computer memory and wastes a large amount of computing resources, which is difficult to meet the needs of low-cost terminals. This paper [1] proposes a fine-grained disease categorization method based on attention network to solve the problem. In "Classification Model", attention mechanism is used to increase identification ability. "Reconstruction-Generation Model" were added during training and the "Classification Model" have to pay more attention to differentiate areas to find differences instead of paying more attention to global features. And adversarial loss was applied to distinguish the generated image from the original image to suppress the noise introduced by the "Discrimination Model" [20]. Due to the feature that "Reconstruction-Generation Model" and "Discrimination Model" are only used in training and do not participate in the operation of inference phase, which cannot increase the complexity of the model. Compared with the traditional classification network, the method of generalization ability enhancement further enhances the identification accuracy. And the method needs less memory but can achieve low performance terminal real-time identification of peach and tomato leaf diseases. And it can be applied in other crop disease identification fields with the similar application scenarios.

The proposed system helps in identification of plant disease and provides remedies that can be used as a defense mechanism against the disease. [1] The database obtained from the Internet is properly segregated and the different plant species are identified and are renamed to form a proper database then obtain test-database which consists of various plant diseases that are used for checking the accuracy and confidence level of the project [2] .Then using training data we will train our classifier and then output will be predicted with optimum accuracy . We use Convolution Neural Network(CNN) [9, 10] which comprises of different layers which are used for prediction. A prototype drone model is also designed which can be used for live coverage of large agricultural fields to which a high resolution camera is attached and will capture images of the plants which will act as input for the software, based of which the software will tell us whether the plant is healthy or not. With our code and training model we have achieved an accuracy level of 78% .Our software gives us the name of the plant species with its confidence level and also the remedy that can be taken as a cure.

Plant diseases are considered one of the main factors influencing food production and minimize losses in production, and it is essential that crop diseases have fast detection and recognition. The recent expansion of deep learning methods has found its application in plant disease detection, offering a robust tool with highly accurate results. In this context, [3] this work presents a systematic review of the literature that aims to identify the state of the art of the use of convolutional neural networks(CNN) [9] in the process of identification and classification of plant diseases, delimiting trends, and indicating gaps. In this sense, we present 121 papers selected in the last ten years with different approaches to treat aspects related

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

to disease detection, characteristics of the data set, the crops and pathogens investigated. From the results of the systematic review, it is possible to understand the innovative trends regarding the use of CNNs [10] in the identification of plant diseases and to identify the gaps that need the attention of the research community.

The identification of plant disease is an imperative part of crop monitoring systems. [14] Computer vision and deep learning (DL) techniques have been proven to be state-of-the-art to address various agricultural problems. This research performed the complex tasks of localization and classification of the disease in plant leaves. In this regard [4], three DL meta-architectures including the Single Shot MultiBox Detector (SSD), Faster Region-based Convolutional Neural Network (RCNN), and Region-based Fully Convolutional Networks (RFCN) were applied by using the TensorFlow object detection framework. [25, 26] All the DL models were trained/tested on a controlled environment dataset to recognize the disease in plant species. Moreover, an improvement in the mean average precision of the best-obtained deep learning architecture was attempted through different state-of-the-art deep learning optimizers. The SSD model trained with an Adam optimizer exhibited the highest mean average precision (mAP) of 73.07%. The successful identification of 26 different types of defected and 12 types of healthy leaves in a single framework proved the novelty of the work. In the future, the proposed detection methodology can also be adopted for other agricultural applications. Moreover, the generated weights can be reused for future real-time detection of plant disease in a controlled/uncontrolled environment.

This paper presents a novel end-to-end DeepPestNet framework for pest recognition and classification [5]. The proposed model has 11 learnable layers, including eight convolutional and three fully connected (FC) layers. We used image rotations techniques to increase the size of the dataset and image augmentations techniques to test the generalizability of the proposed DeepPestNet approach. We used the popular Deng's crops data set to assess the proposed DeepPestNet framework. We used the proposed method to recognize and classify crop pests into 10-class pests, i.e., Locusta migratoria, Euproctis pseudoconspersa strand, chrysochus Chinensis, empoasca flavescens, Spodoptera exigua, larva of laspeyresia pomonella, parasa lepida, acrida cinerea, larva of S. exigua, and L.pomonella types of insects pests. The proposed method achieved optimal accuracy of 100%. [5] We compared the proposed DeepPestNet approach with traditional pre-trained deep learning (DL) models. To verify the general adaptability of this model, we tested the proposed model on the standard Kaggle dataset "Pest Dataset" to recognize nine types of pests: aphids, armyworm, beetle, bollworm, grasshopper, mites, mosquito, sawfly, and stem borer and achieved an accuracy of 98.92%. The proposed model can provide specialists and farmers with immediate and effective aid in recognizing pests, potentially reducing economic and crop yield losses.

2. METHODOLOGY

i) Proposed Work:

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

The proposed system introduces a novel approach named PCA DeepNet, a hybrid framework combining classical Machine Learning through Principal Component Analysis (PCA) and a customized Deep Neural Network. PCA serves as the primary feature extractor, followed by tailored deep neural networks for classification and detection utilizes the Faster Region-Based Convolutional Neural Network (F-RCNN) [4, 36] for effective detection. This project extends its functionality with advanced models, including DenseNet, Xception, and a Voting Classifier for classification, alongside YoloV5 and YoloV8 for detection. The goal is to achieve exceptional accuracy for classification and high precision (mAP of 0.85 or above) for detection. As part of this extension, a user-friendly front end is proposed using the Flask framework, ensuring a seamless and secure testing environment. Flask's flexibility enables the creation of an interactive interface, enhancing user experience. The incorporation of authentication ensures secure access to the system, providing authorized users with a reliable platform to test and interact with the advanced models. This project aims to deliver a comprehensive solution by combining powerful models with an accessible and secure user interface.

ii) System Architecture:

The proposed PCA DeepNet is meticulously designed for performance where each and every step of data preparation and analysis has been optimized for best results. The pre-processed image data is first made to go through an augmentation process using GANs, where the data is refined and made more trainable for further processes. Then the data is processed with a feature extraction technique performed using conventional Principal Component Analysis (PCA). Thereafter, the classification of the data, which is the highlight of PCA DeepNet, is done using a customized CNN Classifier specifically designed to process the data. At last, the classified outputs are detected using a faster region-based CNN [35, 36]. The overall system workflow is shown in Fig.1.

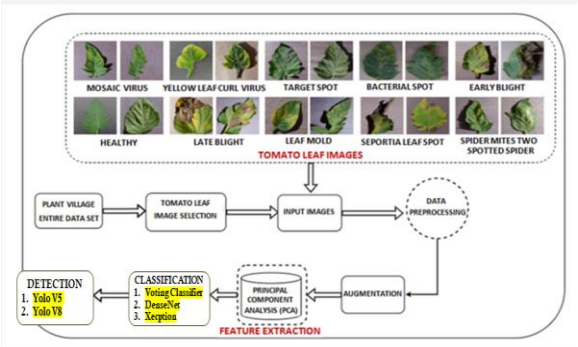


Fig 1 System Architecture

iii) Dataset collection:

For training a Deep Learning model to be able to classify precisely and with high accuracy, an image dataset with proper and balanced image samples is very essential [25]. The larger the dataset size, the more accurate the deep learning model can be obtained. Hence the authors have selected the Plant Village dataset [26] which is an open-source agricultural disease dataset. It is a collection of more than 56000 images divided into 38 classes consisting of 19 crops (apple, grapes, tomato, potato etc.). The dataset consists of high-quality images of leaves in .jpeg format with a width of 5472 pixels and a height of 3648 pixels. Among the 19 crops, the authors have only taken tomatoes into consideration.

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

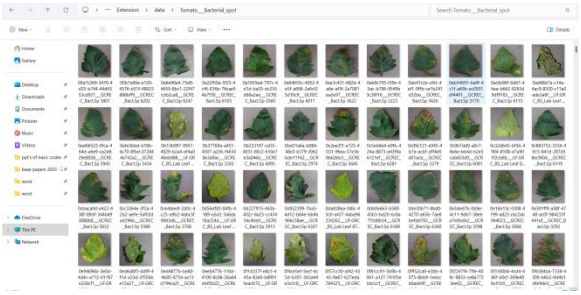


Fig 2 Tomato Dataset

The tomato data is distributed into 10 different classes namely Late_blight, Healthy, Early_blight, Seportia_leaf_ spot, Yellow_leaf_curl_virus, Bacteria_spot, Target_spot, Mosaic_virus, Leaf_mold and Spider_Mites_two_spotted_ sider which consists of a total of 18,128 images of tomato leaves.

iv) Image Processing:

Image processing [11] plays a pivotal role in object detection within autonomous driving systems, encompassing several key steps [18]. The initial phase involves converting the input image into a blob object, optimizing it for subsequent analysis and manipulation. Following this, the classes of objects to be detected are defined, delineating the specific categories that the algorithm aims to identify. Simultaneously, bounding boxes are declared, outlining the regions of interest within the image where objects are expected to be located. The processed data is then converted into a NumPy array, a critical step for efficient numerical computation and analysis.

The subsequent stage involves loading a pre-trained model, leveraging existing knowledge from extensive datasets. This includes reading the network layers of the pre-trained model, containing learned features and parameters vital for accurate object detection. Additionally, output layers are extracted, providing final predictions and enabling effective object discernment and classification.

Further, in the image processing pipeline, the image and annotation file are appended, ensuring comprehensive information for subsequent analysis. The color space is adjusted by converting from BGR to RGB, and a mask is created to highlight relevant features. Finally, the image is resized, optimizing it for further processing and analysis [13, 14]. This comprehensive image processing workflow establishes a solid foundation for robust and accurate object detection in the dynamic context of autonomous driving systems, contributing to enhanced safety and decision-making capabilities on the road.

v) Data Augmentation:

Data augmentation is a fundamental technique in enhancing the diversity and robustness of training datasets for machine learning models, particularly in the context of image processing and computer vision. The process involves three key transformations to augment the original dataset: randomizing the image, rotating the image, and transforming the image.

Randomizing the image introduces variability by applying random modifications, such as changes in brightness, contrast, or color saturation. This stochastic approach helps the model generalize better to unseen data and diverse environmental conditions.

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

Rotating the image involves varying the orientation of the original image by different degrees. This augmentation technique aids in teaching the model to recognize objects from different perspectives, simulating variations in real-world scenarios.

Transforming the image includes geometric transformations such as scaling, shearing, or flipping. These alterations enrich the dataset by introducing distortions that mimic real-world variations in object appearance and orientation.

By employing these data augmentation techniques, the training dataset becomes more comprehensive, allowing the model to learn robust features and patterns. This, in turn, improves the model's ability to generalize and perform effectively on diverse and challenging test scenarios. Data augmentation serves as a crucial tool in mitigating overfitting, enhancing model performance, and promoting the overall reliability of machine learning models, especially in applications like image recognition for autonomous driving systems.

vi) Algorithms:

VGG16 : VGG16, short for Visual Geometry Group 16, is a deep convolutional neural network architecture known for its simplicity and effectiveness .A convolutional neural network is also known as a ConvNet, which is a kind of artificial neural network. A convolutional neural network has an input layer, an output layer, and various hidden layers. VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date [24].

VGG16

```
vgg16=VGG16(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
x1= Flatten()(vgg16.output)
prediction1 = Dense(10, activation='softmax')(x1)
model1 = Model(inputs = vgg16.inputs, outputs = prediction1)
model1.summary()
model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
```

Fig 3 VGG16

InceptionResNetV2; Inception ResNetV2 is a deep neural network architecture that combines elements from the Inception architecture and ResNet (Residual Networks). It is an extension of the original Inception architecture, designed to improve the training and performance of deep convolutional neural networks (CNNs) for image classification and other computer vision tasks.The Inception module, originally introduced in GoogLeNet (InceptionV1), uses multiple parallel convolutional filters of different sizes to capture features at various scales. On the other hand, ResNet introduced residual connections, which allow the network to learn residual mappings, making it easier to train very deep network.

InceptionResNetV2

```
eff = InceptionResNetV2(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
x1= Flatten()(eff.output)
prediction1 = Dense(10, activation='softmax')(x1)
model3 = Model(inputs = eff.inputs, outputs = prediction1)
model3.summary()
model3.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
Model: "model_7"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_8 (InputLayer)         [(None, 128, 128, 3)] 0
_________________________________________________________________________________________________
conv2d_395 (Conv2D)          (None, 63, 63, 32)   864       input_8[0][0]
_________________________________________________________________________________________________
batch_normalization_395 (BatchN (None, 63, 63, 32)   96        conv2d_395[0][0]
_________________________________________________________________________________________________
activation_391 (Activation)  (None, 63, 63, 32)   0         batch_normalization_395[0][0]
_________________________________________________________________________________________________
conv2d_396 (Conv2D)          (None, 61, 61, 32)   9216      activation_391[0][0]
_________________________________________________________________________________________________
batch_normalization_396 (BatchN (None, 61, 61, 32)   96        conv2d_396[0][0]
_________________________________________________________________________________________________
activation_392 (Activation)  (None, 61, 61, 32)   0         batch_normalization_396[0][0]
_________________________________________________________________________________________________
conv2d_397 (Conv2D)          (None, 61, 61, 64)   18432     activation_392[0][0]
```

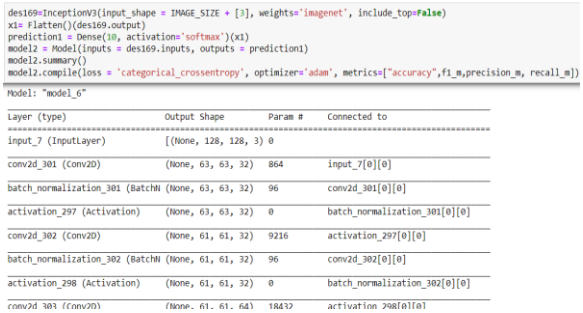
```
hist3 = model3.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set
```

Fig 4 InceptionResnetV2

InceptionV3, InceptionV3 is a convolutional neural network (CNN) architecture designed for image

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

classification and recognition tasks. It is part of the Inception family of architectures developed by Google. InceptionV3 improves upon its predecessor, InceptionV2, by introducing factorized convolutions and batch normalization, resulting in a more efficient and accurate model. It has been widely used in various computer vision applications, demonstrating strong performance in image classification tasks on large datasets.



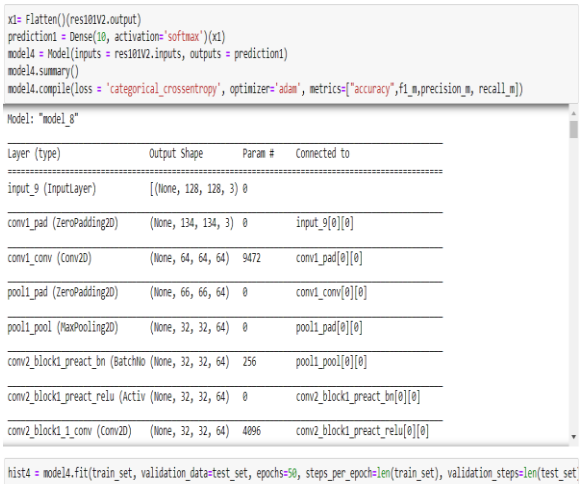
```
des169=InceptionV3(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
x1= Flatten()(des169.output)
prediction1 = Dense(10, activation='softmax')(x1)
model2 = Model(inputs = des169.inputs, outputs = prediction1)
model2.summary()
model2.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
```

Model: "model_6"

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 128, 128, 3)	0	
conv2d_301 (Conv2D)	(None, 63, 63, 32)	864	input_7[0][0]
batch_normalization_301 (BatchN	(None, 63, 63, 32)	96	conv2d_301[0][0]
activation_297 (Activation)	(None, 63, 63, 32)	0	batch_normalization_301[0][0]
conv2d_302 (Conv2D)	(None, 61, 61, 32)	9216	activation_297[0][0]
batch_normalization_302 (BatchN	(None, 61, 61, 32)	96	conv2d_302[0][0]
activation_298 (Activation)	(None, 61, 61, 32)	0	batch_normalization_302[0][0]
conv2d_303 (Conv2D)	(None, 61, 61, 64)	18432	activation_298[0][0]

Fig 5 Inception V3

ResNet152V2, ResNet152V2 is a deep convolutional neural network architecture and an extension of the ResNet (Residual Networks) family. It specifically denotes a ResNet with 152 layers, incorporating the use of residual connections to facilitate the training of very deep networks. The "V2" in the name signifies improvements over the original ResNet, such as the use of bottleneck blocks and other optimizations to enhance both training efficiency and overall performance. ResNet152V2 is often utilized for image classification and other computer vision tasks, particularly when a highly expressive and deep neural network is required.



```
x1= Flatten()(res101V2.output)
prediction1 = Dense(10, activation='softmax')(x1)
model4 = Model(inputs = res101V2.inputs, outputs = prediction1)
model4.summary()
model4.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
```

Model: "model_8"

Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	[(None, 128, 128, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 134, 134, 3)	0	input_9[0][0]
conv1_conv (Conv2D)	(None, 64, 64, 64)	9472	conv1_pad[0][0]
pool1_pad (ZeroPadding2D)	(None, 66, 66, 64)	0	conv1_conv[0][0]
pool1_pool (MaxPooling2D)	(None, 32, 32, 64)	0	pool1_pad[0][0]
conv2_block1_preact_bn (BatchNo	(None, 32, 32, 64)	256	pool1_pool[0][0]
conv2_block1_preact_relu (Activ	(None, 32, 32, 64)	0	conv2_block1_preact_bn[0][0]
conv2_block1_1_conv (Conv2D)	(None, 32, 32, 64)	4096	conv2_block1_preact_relu[0][0]

```
hist4 = model4.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set
```

Fig 6 ResNet152V2

PCA DeepNet, The PCA DeepNet framework is a hybrid approach combining classical machine learning using Principal Component Analysis (PCA) with a customized deep neural network. PCA functions as the primary feature extractor, reducing the dimensionality of the input data, while the deep neural network is tailored for disease detection through classification and pattern recognition. This novel combination aims to enhance accuracy and efficiency in disease detection by synergizing the strengths of both classical and deep learning techniques.

```
# Input Layer
InputL = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3), name="InputLayer")

# Tokenizer
Tokenizer = ConvTokenizer(filters=FILTERS, name="ConvTokenizer")
tokens    = Tokenizer(InputL)


# Positional Embeddings
EL, seq_len = Tokenizer.PE(IMAGE_SIZE)
positions   = tf.range(start=0, limit=seq_len, delta=1)
embedding   = EL(positions)
tokens     += embedding

# Transformer Block
encodings = TransformerBlock(tokens)

# Sequence Pooling
pooled = SeqPool(encodings)

# Classifier
OutputL = Dense(n_classes, activation='softmax', name="OutptLayer")(pooled)

# Model
model = Model(InputL, OutputL, name="CCT")
model.summary()
```

Fig 7 PCA DeepNet

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

KNN, K-Nearest Neighbors (KNN) is a straightforward machine learning algorithm used for classification and regression. It determines the label or value of a new data point based on the majority class or average of its k-nearest neighbors in the feature space. The choice of 'k' is crucial and can impact the algorithm's performance, with KNN being simple to implement but potentially less efficient with larger datasets [29].

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)

knn_acc = accuracy_score(y_test, predictions)
knn_prec = precision_score(y_test, predictions,average='weighted')
knn_rec = recall_score(y_test, predictions,average='weighted')
knn_f1 = f1_score(y_test, predictions,average='weighted')
```

```
storeResults('KNN',knn_acc,knn_prec,knn_rec,knn_f1)
```

Fig 8 KNN

Decision Tree, Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
predictions = dt.predict(X_test)
dt_acc = accuracy_score(y_test, predictions)
dt_prec = precision_score(y_test, predictions,average='weighted')
dt_rec = recall_score(y_test, predictions,average='weighted')
dt_f1 = f1_score(y_test, predictions,average='weighted')
```

```
storeResults('DecisionTree',dt_acc,dt_prec,dt_rec,dt_f1)
```

Fig 9 Decision tree

Random Forest, Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.* As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."**

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(max_depth=2, random_state=0)
RF.fit(X_train, y_train)
predictions = RF.predict(X_test)

rf_prec = precision_score(y_test, predictions,average='weighted')
rf_rec = recall_score(y_test, predictions,average='weighted')
rf_f1 = f1_score(y_test, predictions,average='weighted')
rf_acc = accuracy_score(y_test, predictions)
```

```
storeResults('Random Forest',rf_acc,rf_prec,rf_rec,rf_f1)
```

Fig 10 Random forest

Gradient Boosting, Gradient Boosting is an ensemble technique in machine learning that combines the predictions of weak learners, often decision trees, to build a strong predictive model. It sequentially corrects errors by fitting new learners to the residual errors of the current ensemble. The process involves assigning weights to each learner based on their contribution to error reduction. Gradient Boosting is known for its high predictive accuracy and resilience against overfitting.

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
predictions = gb.predict(X_test)
gb_acc = accuracy_score(y_test, predictions)
gb_prec = precision_score(y_test, predictions,average='weighted')
gb_rec = recall_score(y_test, predictions,average='weighted')
gb_f1 = f1_score(y_test, predictions,average='weighted')
```

```
storeResults('Gradient Boosting',gb_acc,gb_prec,gb_rec,gb_f1)
```

Fig 11 Gradient boosting

Densenet, DenseNet is a convolutional neural network architecture designed for image classification tasks. It features dense connectivity, where each layer receives input from all preceding layers, promoting feature reuse. With bottleneck layers for efficiency and transition layers for dimensionality control, DenseNet achieves parameter efficiency, making it competitive in performance with fewer parameters compared to traditional CNNs.

DenseNet

```
mob1 = DenseNet201(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
x1= Flatten()(mob1.output)
prediction1 = Dense(10, activation='softmax')(x1)
model5 = Model(inputs = mob1.inputs, outputs = prediction1)
model5.summary()
model5.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
zero_padding2d_2 (ZeroPadding2D (None, 134, 134, 3)  0           input_2[0][0]
__________________________________________________________________________________________________
conv1/conv (Conv2D)             (None, 64, 64, 64)   9408        zero_padding2d_2[0][0]
__________________________________________________________________________________________________
conv1/bn (BatchNormalization)   (None, 64, 64, 64)   256         conv1/conv[0][0]
__________________________________________________________________________________________________
conv1/relu (Activation)         (None, 64, 64, 64)   0           conv1/bn[0][0]
__________________________________________________________________________________________________
zero_padding2d_3 (ZeroPadding2D (None, 66, 66, 64)   0           conv1/relu[0][0]
__________________________________________________________________________________________________
pool1 (MaxPooling2D)            (None, 32, 32, 64)   0           zero_padding2d_3[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 32, 32, 64)   256         pool1[0][0]
```

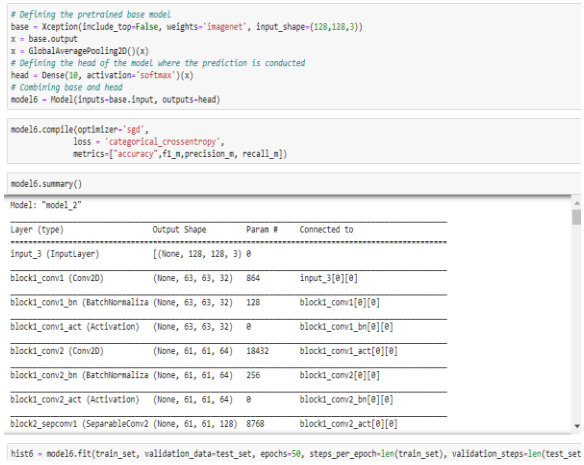
```
hist5 = model5.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set)
```

Fig 12 Densenet

Xecption, Xception is a deep learning architecture introduced by Google that represents an extreme version of the Inception (GoogLeNet) architecture. The name "Xception" stands for "Extreme Inception." It replaces the standard Inception modules with depthwise separable convolutions, which factorize the standard convolution into depthwise and pointwise convolutions. This modification aims to capture complex patterns more efficiently while reducing the number of parameters in the network [8].

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024



```
# Defining the pretrained base model
base = Xception(include_top=False, weights='imagenet', input_shape=(128,128,3))
x = base.output
x = GlobalAveragePooling2D()(x)
# Defining the head of the model where the prediction is conducted
head = Dense(10, activation='softmax')(x)
# Combining base and head
model6 = Model(inputs=base.input, outputs=head)
```

```
model6.compile(optimizer='sgd',
               loss = 'categorical_crossentropy',
               metrics=["accuracy",f1_m,precision_m, recall_m])
```

```
model6.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_3 (InputLayer)         [(None, 128, 128, 3)] 0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 63, 63, 32)   864         input_3[0][0]
_________________________________________________________________
block1_conv1_bn (BatchNormaliza (None, 63, 63, 32)   128         block1_conv1[0][0]
_________________________________________________________________
block1_conv1_act (Activation)   (None, 63, 63, 32)   0           block1_conv1_bn[0][0]
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 61, 61, 64)   18432       block1_conv1_act[0][0]
_________________________________________________________________
block1_conv2_bn (BatchNormaliza (None, 61, 61, 64)   256         block1_conv2[0][0]
_________________________________________________________________
block1_conv2_act (Activation)   (None, 61, 61, 64)   0           block1_conv2_bn[0][0]
_________________________________________________________________
block2_sepconv1 (SeparableConv2 (None, 61, 61, 128)  8768        block1_conv2_act[0][0]
```

```
hist6 = model6.fit(train_set, validation_data=test_set, epochs=50, steps_per_epoch=len(train_set), validation_steps=len(test_set
```

Fig 13 Xception

Voting Classifier, A voting classifier is a machine learning model that gains experience by training on a collection of several models and forecasts an output (class) based on the class with the highest likelihood of becoming the output. To forecast the output class based on the largest majority of votes, it averages the results of each classifier provided into the voting classifier. The concept is to build a single model that learns from various models and predicts output based on their aggregate majority of votes for each output class, rather than building separate specialized models and determining the accuracy for each of them.

Voting Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
clf1 = DecisionTreeClassifier()
clf2 = RandomForestClassifier()
clf3 = GradientBoostingClassifier()
eclf1 = VotingClassifier(estimators=[('dt', clf1),('rf', clf2), ('gb', clf3)], voting='soft')
eclf1.fit(X_train, y_train)

predictions = eclf1.predict(X_test)

vot_acc = accuracy_score(y_test, predictions)
vot_prec = precision_score(y_test, predictions,average='weighted')
vot_rec = recall_score(y_test, predictions,average='weighted')
vot_f1 = f1_score(y_test, predictions,average='weighted')
```

```
storeResults('Voting Classifier',vot_acc,vot_prec,vot_rec,vot_f1)
```

Fig 14 Voting classifier

Faster RCNN, Faster R-CNN is a deep learning model designed for object detection tasks in computer vision. It introduces a region proposal network (RPN) to efficiently propose potential object regions within an image. The RPN generates region proposals, and these proposals are then used to predict bounding boxes and class probabilities for objects. The model is known for its accuracy and ability to handle complex scenes with multiple objects. Faster R-CNN is employed in the project for its proven accuracy in object detection and its efficient two-stage architecture, featuring a region proposal network, which is crucial for accurately identifying and localizing diseases in tomato plant images [4].

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

Fig 15 Faster RCNN

YOLOv5 (You Only Look Once version 5):YOLOv5 is an object detection model that operates by dividing an image into a grid and making predictions for bounding boxes and class probabilities directly within each grid cell in a single pass through the network. It is an evolution of the YOLO series, designed to provide a balance between speed and accuracy, making it well-suited for real-time object detection tasks.

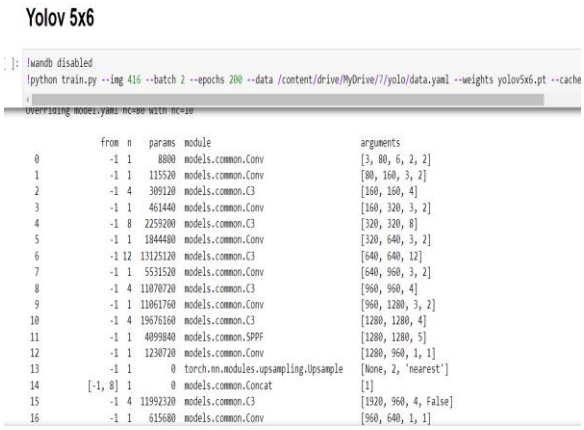


Fig 16 YOLOV5

YOLOv8 (You Only Look Once version 8):YOLOv8 is a cutting-edge computer vision model renowned for its excellence in object detection, segmentation, and pose estimation. As an evolution of the YOLO legacy, it delivers state-of-the-art performance through a streamlined design and innovative features. YOLOv8 is characterized by its speed, accuracy, and user-friendly nature, making it suitable for a wide range of applications, including real-time object tracking, image classification, and pose recognition.

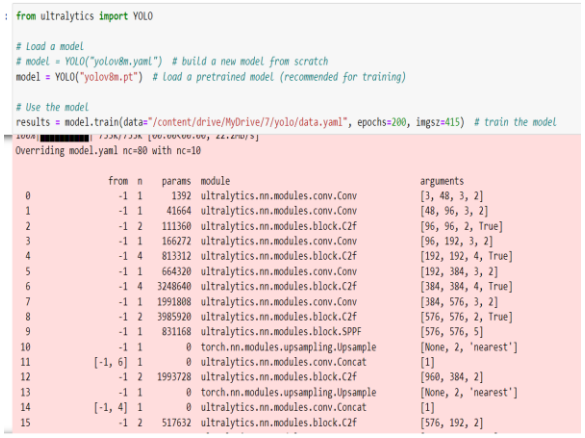


Fig 17 YOLOV8

3. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issuse 2.May 2024

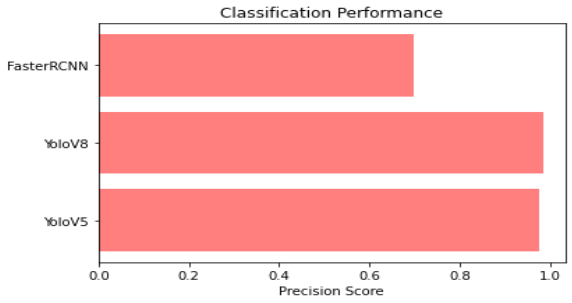


Fig 18 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

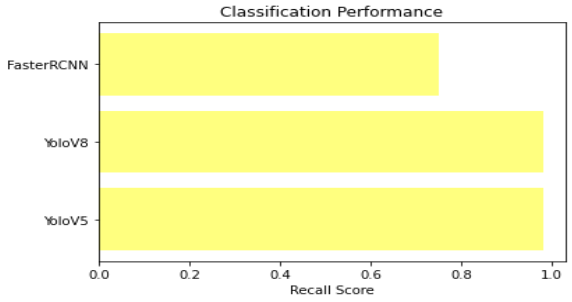


Fig 19 Recall comparison graph

mAP: Mean Average Precision (MAP) is a ranking quality metric. It considers the number of relevant recommendations and their position in the list. MAP at K is calculated as an arithmetic mean of the Average Precision (AP) at K across all users or queries.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

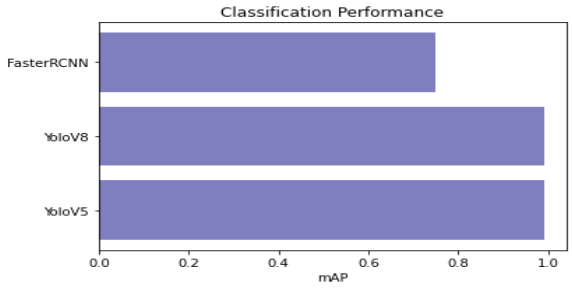


Fig 20 mAP comparison graph

Algorithms	Accuracy	Precision	Recall	F1 - score
PCA DeepNet	0.953	0.885	1.000	0.939
VGG16	0.957	0.959	0.956	0.958
InceptionV3	0.963	0.963	0.963	0.963
InceptionResNetV2	0.945	0.945	0.945	0.945
ResNet152V2	0.950	0.956	0.945	0.950
KNN	0.554	0.600	0.554	0.542
DecisionTree	0.469	0.473	0.469	0.471
Random Forest	0.353	0.351	0.353	0.209
Gradient Boosting	0.774	0.769	0.774	0.766
Extension DenseNet	0.983	0.984	0.982	0.983
Extension Xception	1.000	1.000	1.000	1.000
Exetsnion Voting Classifier	1.000	1.000	1.000	1.000

Algorithms	Precision	Recall	F1 - score
YoloV5	0.977	0.98	0.99
YoloV8	0.984	0.98	0.99
Faster RCNN	0.698	0.75	0.75

Fig 21 Performance Evaluation table

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

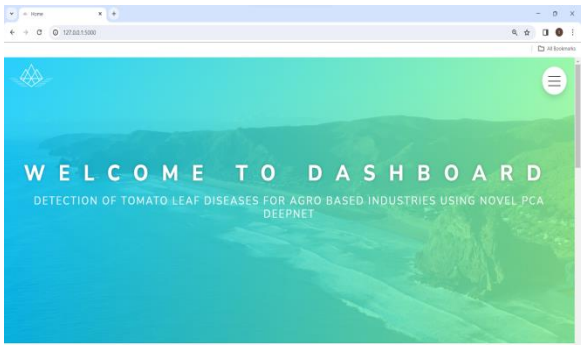


Fig 22 Home page

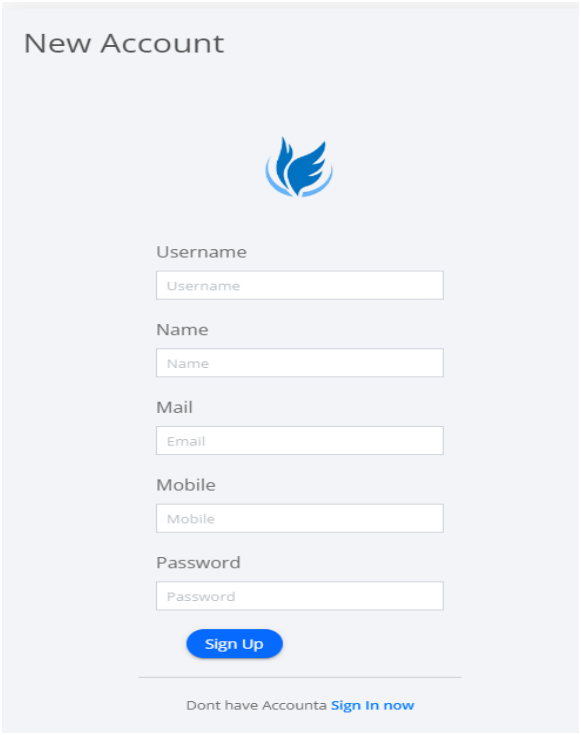


Fig 23 Registration page

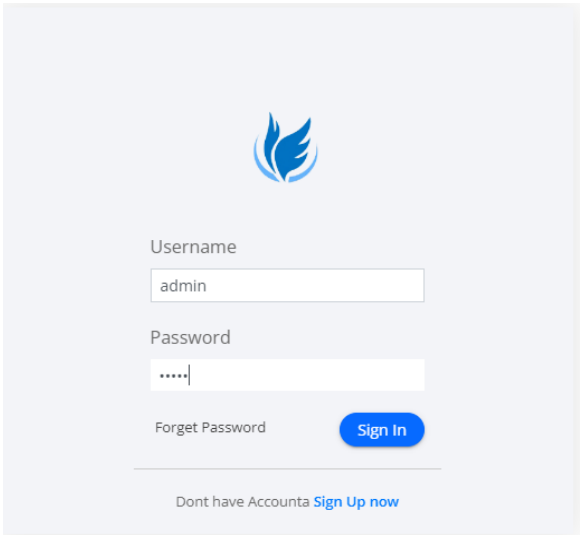


Fig 24 Login page

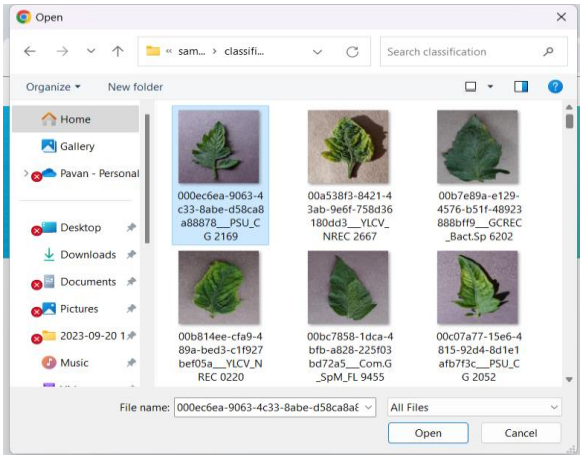
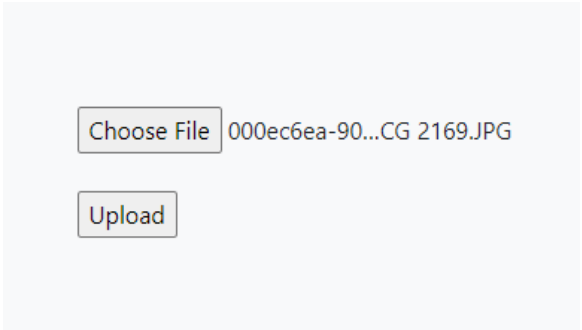


Fig 25 Input image folder



INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 14,Issuse 2.May 2024

Fig 26 Upload input image

Fig 27 Predict result for given input

4. CONCLUSION

The project seamlessly combines deep learning (PCA DeepNet, VGG16, InceptionResNetV2) with traditional machine learning models (KNN, Decision Tree, Random Forest) to create a comprehensive solution for precise tomato leaf disease detection [6]. Meticulous image preprocessing, including re-scaling and transformation, optimizes data for model training, enhancing adaptability to diverse agricultural conditions. Tailored model architectures, such as VGG16 and InceptionResNetV2, showcase a strategic approach to capturing both global and intricate features, ensuring robust disease identification. The other algorithms has demonstrated exceptional performance, showcasing the effectiveness of the diversified model ensemble for classification and state-of-the-art YoloV5 and YoloV8 models for precise object detection. Rigorous testing in the user-friendly front end, with real-world feature values, further validates its reliability and practical applicability in accurately identifying and localizing tomato leaf diseases [4, 6, 10]. Integration of Flask and SQLite enables user-friendly interactions, allowing farmers to easily sign up, upload images, and receive prompt analyses for informed decision-making. 6. By leveraging AI for early disease detection, the project empowers precision agriculture, offering a proactive solution that preserves crop quality and fosters sustainable farming practices.

5. FUTURE SCOPE

Automation of the proposed system into a real-time system can directly benefit farmers and others associated with agriculture, providing timely detection and prevention of tomato leaf diseases [4]. Further optimization of the system can be explored by experimenting with different pre-trained deep learning models and hyperparameters, comparing the efficiency of the proposed PCA DeepNet classifier . Integration of the system with other agricultural technologies and platforms can enhance its usability and accessibility, allowing for seamless integration into existing agricultural practices. Expansion of the system to detect and classify diseases in other crops can be considered, broadening its applicability and impact in the field of agriculture.

REFERENCES

[1] Y. Wu, X. Feng, and G. Chen, ‘‘Plant leaf diseases fine-grained categorization using convolutional neural networks,’’ IEEE Access, vol. 10, pp. 41087–41096,

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

2022. [2] M. Adnan, K. Ali, G. Drushti, and C. Tejal, ‘‘Plant disease detection using CNN & remedy,’’ Int. J. Adv. Res. Electr., Electron. Instrum. Eng., vol. 8, no. 3, pp. 622–626, 2019.

[3] A. Abade, P. A. Ferreira, and F. de Barros Vidal, ‘‘Plant diseases recognition on images using convolutional neural networks: A systematic review,’’ Comput. Electron. Agricult., vol. 185, Jun. 2021, Art. no. 106125.

[4] M. H. Saleem, S. Khanchi, J. Potgieter, and K. M. Arif, ‘‘Image-based plant disease identification by deep learning meta-architectures,’’ Plants, vol. 9, no. 11, p. 1451, Oct. 2020.

[5] N. Ullah, J. A. Khan, L. A. Alharbi, A. Raza, W. Khan, and I. Ahmad, ‘‘An efficient approach for crops pests recognition and classification based on novel DeepPestNet deep learning model,’’ IEEE Access, vol. 10, pp. 73019–73032, 2022.

[6] S. Ahmed, M. B. Hasan, T. Ahmed, M. R. K. Sony, and M. H. Kabir, ‘‘Less is more: Lighter and faster deep neural architecture for tomato leaf disease classification,’’ IEEE Access, vol. 10, pp. 68868–68884, 2022.

[7] E. Elfatimi, R. Eryigit, and L. Elfatimi, ‘‘Beans leaf diseases classification using MobileNet models,’’ IEEE Access, vol. 10, pp. 9471–9482, 2022.

[8] L. Aversano, M. L. Bernardi, M. Cimitile, M. Iammarino, and S. Rondinella, ‘‘Tomato diseases classification based on VGG and transfer learning,’’ in Proc. IEEE Int. Workshop Metrol. Gricult. Forestry (MetroAgriFor), Nov. 2020, pp. 129–133.

[9] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, ‘‘Deep neural networks based recognition of plant diseases by leaf image classification,’’ Comput. Intell. Neurosci., vol. 2016, pp. 1–11, May 2016.

[10] E. Ozbilge, M. K. Ulukok, O. Toygar, and E. Ozbilge, ‘‘Tomato disease recognition using a compact convolutional neural network,’’ IEEE Access, vol. 10, pp. 77213–77224, 2022.

[11] H. Ajra, M. K. Nahar, L. Sarkar, and M. S. Islam, ‘‘Disease detection of plant leaf using image processing and CNN with preventive measures,’’ in Proc. Emerg. Technol. Comput., Commun. Electron. (ETCCE), Dec. 2020, pp. 1–6.

[12] M. Chohan, A. Khan, R. Chohan, S. Hassan, and M. Mahar, ‘‘Plant disease detection using deep learning,’’ Int. J. Recent Technol. Eng., vol. 9, no. 1, pp. 909–914, 2020.

[13] A. Rao and S. B. Kulkarni, ‘‘A hybrid approach for plant leaf disease detection and classification using digital image processing methods,’’ Int. J. Electr. Eng. Educ., Oct. 2020, Art. no. 002072092095312, doi: 10.1177/0020720920953126.

[14] M. K. Singh, S. Chetia, and M. Singh, ‘‘Detection and classification of plant leaf diseases in image processing using MATLAB,’’ Int. J. Life Sci. Res., vol. 5, no. 4, pp. 120–124, 2017.

[15] A. Patel and B. Joshi, ‘‘A survey on the plant leaf disease detection techniques,’’ Int. J. Adv. Res. Comput. Commun. Eng., vol. 6, no. 1, pp. 229–231, 2017.

ISSN 2229-6107 www.ijpast.in
Vol 14,Issue 2.May 2024

[16] J. Gui, L. Hao, Q. Zhang, and X. Bao, ‘‘A new method for soybean leaf disease detection based on modified salient regions,’’ Int. J. Multimedia Ubiquitous Eng., vol. 10, no. 6, pp. 45–52, 2015.

[17] S. Pavithra, A. Priyadharshini, V. Praveena, and T. Monika, ‘‘Paddy leaf disease detection using SVM classifier,’’ Int. J. Commun. Comput. Technol., vol. 3, no. 1, pp. 16–20, 2015.

[18] Y. M. Oo and N. C. Htun, ‘‘Plant leaf disease detection and classification using image processing,’’ Int. J. Res. Eng., vol. 5, no. 9, pp. 516–523, Nov. 2018.

[19] D. Ashourloo, H. Aghighi, A. A. Matkan, M. R. Mobasheri, and A. M. Rad, ‘‘An investigation into machine learning regression techniques for the leaf rust disease detection using hyperspectral measurement,’’ IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 9, no. 9, pp. 4344–4351, Sep. 2016.

[20] H. Nazki, S. Yoon, A. Fuentes, and D. S. Park, ‘‘Unsupervised image translation using adversarial networks for improved plant disease recognition,’’ Comput. Electron. Agricult., vol. 168, Jan. 2020, Art. no. 105117.

[21] Q. Zeng, X. Ma, B. Cheng, E. Zhou, and W. Pang, ‘‘GANs-based data augmentation for citrus disease severity detection using deep learning,’’ IEEE Access, vol. 8, pp. 172882–172891, 2020.

[22] Y. Zhao, Z. Chen, X. Gao, W. Song, Q. Xiong, J. Hu, and Z. Zhang, ‘‘Plant disease detection using generated leaves based on DoubleGAN,’’ IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 19, no. 3, pp. 1817–1826, May/Jun. 2021.

[23] L. Li, S. Zhang, and B. Wang, ‘‘Plant disease detection and classification by deep learning—A review,’’ IEEE Access, vol. 9, pp. 56683–56698, 2021.

[24] M. Brahimi, S. Mahmoudi, K. Boukhalfa, and A. Moussaoui, ‘‘Deep interpretable architecture for plant diseases classification,’’ in Proc. Signal Process., Algorithms, Architectures, Arrangements, Appl. (SPA), Sep. 2019, pp. 111–116.

[25] J. G. A. Barbedo, ‘‘Factors influencing the use of deep learning for plant disease recognition,’’ Biosyst. Eng., vol. 172, pp. 84–91, Aug. 2018.

[26] PlantVillage Dataset. Accessed: Aug. 2, 2022. [Online]. Available: https://www.kaggle.com/abdallahalidev/plantvillage-dataset

[27] F. Harrou, M. N. Nounou, H. N. Nounou, and M. Madakyaru, ‘‘Statistical fault detection using PCA-based GLR hypothesis testing,’’ J. Loss Prevention Process Ind., vol. 26, no. 1, pp. 129–139, 2013.

[28] A. M. Dawud, K. Yurtkan, and H. Oztoprak, ‘‘Application of deep learning in neuroradiology: Brain haemorrhage classification using transfer learning,’’ Comput. Intell. Neurosci., vol. 2019, pp. 1–12, Jun. 2019.

[29] F. Mohameth, C. Bingcai, and K. A. Sada, ‘‘Plant disease detection with deep learning and feature

ISSN 2229-6107 www.ijpast.in
Vol 14,Issuse 2.May 2024

extraction using plant village,'' J. Comput. Commun., vol. 8, no. 6, pp. 10–22, 2020.

[30] A. Rehman, S. Naz, M. I. Razzak, F. Akram, and M. Imran, ''A deep learning-based framework for automatic brain tumors classification using transfer learning,'' Circuits, Syst., Signal Process., vol. 39, no. 2, pp. 757–775, Feb. 2020.

[31] Y. H. Bhosale and K. Sridhar Patnaik, ''IoT deployable lightweight deep learning application for COVID-19 detection with lung diseases using RaspberryPi,'' in Proc. Int. Conf. IoT Blockchain Technol. (ICIBT), May 2022, pp. 1–6.

[32] A. Tripathy, A. Agrawal, and S. K. Rath, ''Classification of sentiment reviews using n-gram machine learning approach,'' Expert Syst. Appl., vol. 57, pp. 117–126, Sep. 2016.

Access, vol. 7, pp. 59069–59080, 2019

[33] V. V. Srinidhi, A. Sahay, and K. Deeba, ''Plant pathology disease detection in apple leaves using deep convolutional neural networks: Apple leaves disease detection using EfficientNet and DenseNet,'' in Proc. 5th Int. Conf. Comput. Methodolog. Commun. (ICCMC), Apr. 2021, pp. 1119–1127.

[34] E. Harte, ''Plant disease detection using CNN,'' ResearchGate, Sep. 2020, doi: 10.13140/RG.2.2.36485.99048.

[35] Y. Toda and F. Okura, ''How convolutional neural networks diagnose plant disease,'' Plant Phenomics, vol. 2019, pp. 1–14, Mar. 2019.

[36] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, ''Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks,'' IEEE