



## RESEARCH ARTICLE

## OPEN ACCESS

## AN INTEGRATED DEEP LEARNING FRAMEWORK FOR TOMATO LEAF DISEASE CLASSIFICATION AND DETECTION

Yuvaraj Singh Tagore<sup>1</sup>, Manna Sheela Rani Chetty<sup>2</sup><sup>1,2</sup>Department of CSE, Koneru Lakshmaiah Foundation, Vaddeswaram, Andhra Pradesh, India<sup>1</sup><http://orcid.org/0009-0004-2190-6159>, <sup>2</sup><http://orcid.org/0000-0002-1378-6917>E-mail: [yuvarajtagore@gmail.com](mailto:yuvarajtagore@gmail.com), [sheelarani\\_cse@kluniversity.in](mailto:sheelarani_cse@kluniversity.in)

## ARTICLE INFO

*Article History*

Received: February 20, 2026

Reviewed: March 20, 2026

Accepted: May 15, 2026

Published: September 30, 2026

*Keywords:*Machine Learning,  
Computer Vision,  
Convolutional Neural Networks,  
Quicker Region-Based CNN,  
Generative Adversarial  
Networks.

## ABSTRACT

In this study, there is better DL and computer vision techniques or strategies to optimize tomato leaf ailment detection and growth agricultural productiveness. The PCA DeepNet architecture represents a modern approach for shooting facts complexity and disease patterns with the aid of combining conventional ML with a customized DNN. The research successfully categorizes wholesome and diseased tomato leaves utilizing the "faster region-based Convolutional Neural network (F-RCNN)," exhibiting excessive common precision and great classification accuracy. The findings imply that the PCA DeepNet architecture surpasses opportunity approaches, doubtlessly providing good sized benefits to agricultural industries with the aid of accurately and reliably detecting ailments in tomato leaves. This task complements its capability with the aid of incorporating elaborate models inclusive of DenseNet and Xception for classification, at the side of a voting Classifier, in addition to YoloV5 and YoloV8 for detection. The goals are a type accuracy of 0.85 or more and a detection precision of 0.85 or more.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

## I. INTRODUCTION

For the huge majority of countries, agriculture has existed for a tremendous length. Its long-term viability is crucial. For the reason that India's economy predominantly relies on agriculture, a full-size bite of the populace is linked to this zone. The export of agricultural merchandise significantly contributes to the nation's GDP. Have to current tendencies hold, the global population is projected to surpass 10 billion by 2050, requiring a 70% growth in agricultural production. As the world population has elevated, so has the call for food and different agricultural products. The important thing to improving agricultural output is living inside the optimization of modern land use, as no additional arable land is available. In several locations of India, conventional agriculture, which calls for large time and effort, is essentially carried out manually. Preserving plants without pathogens is an important element of agriculture, among different elements. Because of human errors and the inherent boundaries of manual hard work, this approach is susceptible to inaccuracies, which can cause catastrophic consequences, which includes the entire destruction of crops. Farmers are contemplating artificial intelligence (AI) solutions, due to the fact they can't have the funds for to overlook this challenge. Artificial intelligence (AI) offers a more efficient method for analyzing the daily troubles faced through farmers. Pest infestations, which diminish crop pleasant, are among the maximum widespread worries.

In a bid to deal with such pests, farmers are seeking modern system capable of detecting sicknesses as a result of their infections. The combination of artificial intelligence and a computer vision can provide an excellent opportunity in such matters. The ability to utilize real-time data to carry out the roles of forecasting and identifying pests and diseases before they can cause significant harm to the property is one of the most promising AI properties. The ultimate goal of increasing AI-controlled automatic frameworks is to reduce the susceptibility to pest attacks despite maintaining the quality of output. The identification of plant diseases using Deep learning approaches that involve identification and detection has become a necessary aspect in order to screen and analyze the production of every plant species. DL methods are better than classical classification networks in the context of real time identity of plant leaf diseases [1]. The existing method of reputation of the disease is based on visual observation, which is time-consuming and labor-intensive; the miles the relevant element of the present trends on computer vision-aided systems, which effectively deal with a plethora of plant diseases [2]. Deep learning has thus been adopted to deal with this problem, which is a strong method that delivers higher accuracy in identifying disorders [3].

The suggested algorithm of identifying tomato leaf ailments incorporates both DL and ML algorithms. This model demonstrates how the photos of tomato leaves can be categorized into healthy and ill groups, with the help of using the photographs as different ailment-detecting schemes. This comprehensive technique is based on a unique model known as PCA DeepNet, which employs Deep Neural Networks, particularly Convolutional Neural network architectures, and principal element analysis. To begin with, principal component analysis (PCA) is carried out for feature extraction, accompanied by the software of superior DNN for detection and classification. To facilitate the improvement of smart structures for sickness identification in tomato leaves, convolutional DNN are broadly speaking chosen for their ability to enhance type and decrease processing fees. The detection technique employs a "faster region based CNN (F-RCNN) along with a single shot detector (SSD)." In contrast to SSD, F-RCNN executes its detection approaches in two steps, leading to enhanced detection accuracy [4].

## II. LITERATURE SURVEY

As a result of minimal inter-class version in leaf ailment images and enormous intra-class version, it's far vital to capture the attributes of the goal's instantaneous surroundings [1] [4-7]. Furthermore, meeting desires with low-cost terminals is hard because to the enormous reminiscence and useful resource consumption of sophisticated networks. This research affords a method utilizing an attention network for complete sickness classification [8] [9]. "Category model" utilizes an interest mechanism to enhance identity accuracy [10] [11]. The "classification model" must prioritize differentiating areas to pick out variances as opposed to concentrating on usual features, given that "Reconstruction-generation models" had been incorporated into the training technique [12] [13]. The "Discrimination model" delivered noise; subsequently, adversarial loss was employed to differentiate the generated image from the original image [14-17]. making use of traits along with "Reconstruction-technology version" and "Discrimination version" during schooling does no longer enhance the model's complexity, as they are inapplicable non the inference phase. The approach to enhance generalizability consequences in superior identity accuracy relative to traditional classification networks. Moreover, the approach enables real-time detection of illnesses affecting peach and tomato leaves on low-performance devices while necessitating minimal memory usage. It additionally has similar potential applications in several domains of agricultural disorder identification. The proposed approach serves as a preventive tool, facilitating the detection of plant illnesses and supplying suitable treatment options for these situations. Facts schooling extends until October 2023. Upon appropriately categorizing and renaming numerous plant species in the received database from the internet, we boost to collect a test database that encompasses a number plant illnesses to confirm the project's precision and self-assurance level [2]. Subsequently, we will teach our classifier utilizing the training data, after which we might also get the most unique predictions concerning the output.

To facilitate predictions, we make use of a Convolutional Neural network (CNN) [18] [19]. This network incorporates several degrees. Moreover, we have created a drone model capable of real-time monitoring of extensive agricultural fields. This drone is ready with a high-resolution digital camera able to capturing images of plants, as a way to then be analyzed through a laptop to assess their health repute. We've executed an accuracy price of 78% the use of our code and training model. Our algorithm no longer best recognizes plant species but also resources its self-belief stage and ability therapeutic characteristics. The activate analysis and identity of crop diseases are essential, as plant sicknesses significantly have an effect on food production and lead to yield losses. An effective tool with very correct consequences has just emerged for the identification of plant sicknesses, attributed to improvements in deep learning algorithms. This have a look at gives a literature overview targeted on recent improvements inside the prognosis and type of plant sicknesses with convolutional neural networks (CNNs) [9], emphasizing existing deficiencies, constraining tendencies, and presenting contextual records. To do that, we present 121 articles decided on during the last decade that hire various methodologies to address sickness identity, dataset characteristics, crops, and pathogens. Innovative methods for identifying plant sicknesses through convolutional neural networks (CNNs) [10] and understanding gaps necessitating more studies may be inferred from the consequences of the systematic evaluation. Crop tracking structures are useless without the functionality to perceive plant diseases [14]. Computer vision and DL techniques had been employed to create superior solutions addressing numerous agricultural demanding situations. This studies performed the identification and categorization of ailments in plant leaves. In this context, three DL meta-architectures had been implemented the use of the TensorFlow object detection framework: SSD, quicker RCNN, and RFCN.

A dataset created to duplicate a managed environment was utilized to teach and determine all deep learning models capable of figuring out plant illnesses [14]. Attempts had been undertaken to rent numerous sophisticated deep learning optimizers to enhance the common mean precision of the top-performing DL architecture. The SSD version, supplemented with an Adam optimizer, achieved a mAP of 74.07%, surpassing all others. The uniqueness of the approach became highlighted by using its capacity to discover 26 types of defective leaves and 12 kinds of healthy leaves within a unified framework. The proposed detection technique is viable for potential agricultural packages. Furthermore, generated weights can be repurposed for prospective applications in both regulated and unregulated settings, facilitating the spark off identification of plant diseases [20-22]. This paper gives a novel DeepPestNet architecture designed for thorough pest identity and category [23-25]. The proposed model includes 11 trainable layers, along with three fully connected (FC) layers and eight convolutional layers. We greater the dataset with the aid of the utility of image rotation strategies and evaluated the generalizability of the DeepPestNet method utilizing image augmentation techniques. To perform this, we assessed the proposed DeepPestNet architecture the usage of the hooked up Deng's plants dataset. Using the proposed method, we determined and categorized ten wonderful species of agricultural pests: *Locusta migratoria*, *Euproctis pseudoconspersa* Strand, *Chrysochus chinensis*, *Empoasca flavescens*, *Spodoptera exigua*, *Parasa lepida*, *Acrida cinerea*, *L. pomonella*, and the *Laspeyresia pomonella* larva. The goal of the suggested method was to get a perfect score of 100%. We compared the suggested DeepPestNet method with older deep learning models that were trained ahead of time. We used the common "Pest Dataset" from Kaggle to test the suggested model and find nine types of pest: aphids, armyworms, beetles, bollworms, grasshoppers, mites, mosquitoes, sawflies, and stem borers. The model's normal adaptability was shown by the 98.92% accuracy rate. The suggested way can help both farmers and experts figure out what kind of pest they are quickly and efficiently. This could help keep farmers from losing money and keep agriculture productive.

### III. METHODOLOGY

#### i) Proposed Work:

Presenting principal component analysis (PCA) DeepNet, a hybrid framework that integrates a tailored Deep Neural network with classical ML using principal component analysis (PCA), introduces a progressive technique to ML. utilizing the F-RCNN [4], specialized deep neural networks are applied for type following principal component analysis (PCA), which serves as the number one function extractor. This project enhances its functionality by incorporating complicated models like as DenseNet, Xception, and a voting Classifier for type, with YoloV5 and YoloV8 for detection. The classification accuracy should be 0.85 or higher and the detection precision should be 0.85 or higher. Flask framework is proposed due to its human friendly the front end, and ensuring a comfortable and smooth testing environment somewhere in the course of this growth. The adaptability of flask allows the easy development of an interactive interface, which in turn enhances the overall enjoyment of consumers with the legal customers to evaluate and transact with advanced models through the protection of the system access. The goal of this challenge is to create a complete solution by integrating robust models with a user-friendly and secure interface.

#### ii) System Architecture:

Every level of facts practice and analysis for the proposed PCA DeepNet has passed through meticulous optimization to guarantee most desirable overall performance. Before further processing, pre-processed image information undergoes augmentation via using GANs. This system enhances facts and increases its trainability. The subsequent step is to do classic principal component analysis (PCA) at the information to extract functions. Data category, the primary power of PCA DeepNet, is then finished the use of a customized CNN Classifier designed with the information's characteristics in consideration. Outcomes are in the end received through an expedited region-based CNN. Figure 1 illustrates the comprehensive process of the system.

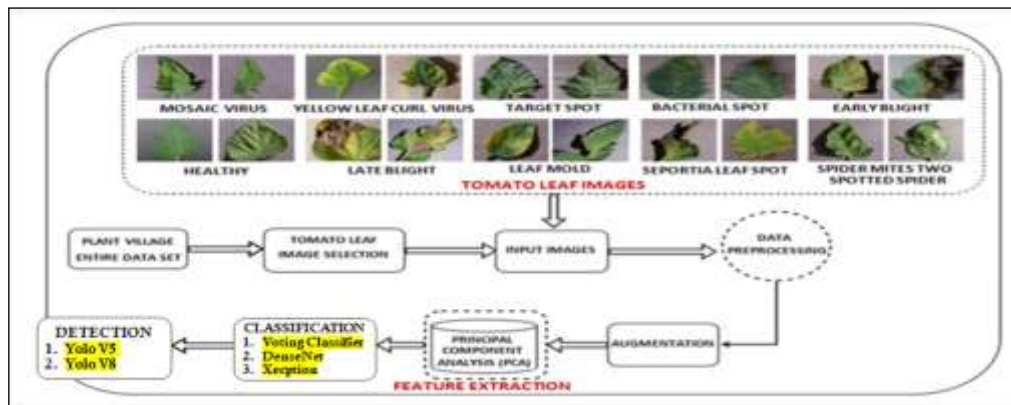


Figure 1: System Architecture.

Source: Authors, (2026).

#### iii) Dataset collection:

A well-balanced image collection is essential for training a DL model to make certain correct and specific categorization [26]. An improved dataset can bring about an extra particular deep learning model. This explains why authors have applied the open-source Plant Village dataset [27] [26] for his or her agricultural disorder investigation. Approximately 56,000 images are labeled into 38 classifications, encompassing 19 wonderful vegetation (e.g., apples, grapes, tomatoes, potatoes, etc.). The leaf images in the series are of high great and saved as .jpeg files with measurements of 5472 pixels in width and 3648 pixels in top. Writers have exclusively focused on tomatoes among nineteen plants. It is seen in figure 2.

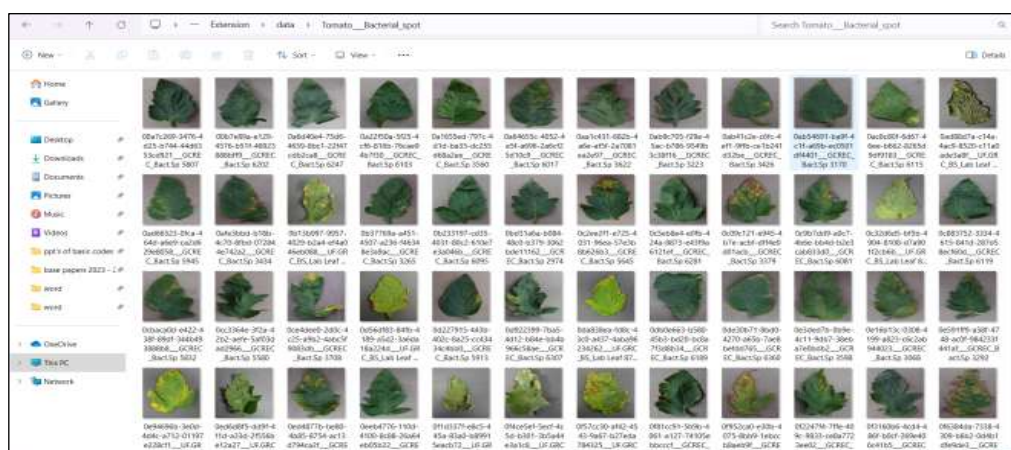


Figure 2: Tomato Dataset.

Source: Authors, (2026).

The research collects 18,128 images from tomato leaves in ten different classes which include overdue Blight and healthful and Early Blight and Septoria leaf spot and Bacterial spot and goal spot and Mosaic virus and Leaf mildew and -spotted spider mites.

#### iv) Image Processing:

Object detection in self-reliant driving structures relies heavily on image processing, which encompasses numerous important strategies [18]. The preliminary phase includes improving the enter image through evaluation and adjustment via reworking it right into a blob object. Eventually, the algorithm's target classes are delineated with the aid of identifying the varieties of items to be identified. Simultaneously, we delineate regions of interest in the image where we wish for items to take place through specifying bounding boxes. An important step in powerful numerical evaluation and computing is converting processed records into a NumPy array. The subsequent phase is loading a pre-trained model that leverages earlier information from large datasets. Such access to the network layers of a pre-trained model is an essential problem of this type as it contains the parameters and learned features that will be essential to the accurate detection of items. Finally, extracted output layers give predictions that enable effective item popularity and type. The image and annotation documents are incorporated into the image processing pipeline in order to guarantee comprehensive information to be used in the future when evaluating the image. A mask is created in order to highlight important information throughout the conversion of the coloration space between BGR to RGB. The final level is to reduce the size of the image taking into consideration the improvements of processing and analysis [13] [14]. This entire picture application infrastructure adds to street safety and decision-making capacities by adopting arranging a foundation of reliable item popularity within the evolving setting of autonomous using arrangements.

#### v) Data Augmentation:

Data augmentation is a very important method of generating more diverse and stronger training data of machine learning models that are to be used in computer vision and image processing. Three main techniques that are carried out to add extra work to the basic dataset are image randomization, rotation, and transformation. The randomized image produces versions by introducing arbitrary alterations which can be variations in contrast, saturation or brightness. The version is capable of generalizing additional efficaciously to sparkling information and many inherent circumstances due to its stochastic approach. Rotating an image is the changing of the position of the original image in the form of different levels. This augmentation methodology conditions the version to be able to perceive objects in different angles by simulating changes that are seen in real-life situations. Image manipulation makes use of geometric operations such as scaling, shearing and flipping. Those adjustments ornament the dataset by relying on incorporating distortions that are variations of the arrival and orientation of objects of the real world. The model can broaden robust capabilities and patterns because the training dataset will increase thru the application of these information augmentation strategies. Therefore, the version's capacity for generalization and its efficacy on difficult take a look at situations are stepped forward. In programs like image reputation for self-sufficient driving systems, data augmentation is an important technique for reducing overfitting, enhancing model performance, and growing the general reliability of machine learning models.

#### vi) Algorithms:

**VGG16:** VGG16, representing visible Geometry group 16, is an outstanding and efficient structure for deep convolutional neural networks. A ConvNet is an artificial neural network that employs convolutional layers to get new insights. A convolutional neural community comprises a couple of hidden layers along enter and output layers. The VGG16 Convolutional Neural network (CNN) is often taken into consideration a premier model within the realm of computer vision. Figure 3 illustrates the VGG16 model employed for class, incorporating ImageNet weights, a further Flatten layer, and a dense layer, and constructed utilizing accuracy, precision, recall, and F1-score metrics.

**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])

```

Figure 3: VGG16.  
Source: Authors, (2026).

**InceptionResNetV2;** Initial film ResNetV2, which merges characteristics of Inception framework & ResNet, is a type of DNN. Considering applications in computer vision & image type, it is an extension of initial Inception architecture designed to improve CNN training & efficacy. To acquire facts at various scales, Inception module makes use of multiple simultaneous convolutional filters of different sizes. This approach become first of all carried out in GoogLeNet (InceptionV1). Training extremely deep networks has end up markedly simpler way to ResNet, which added residual connections that allow network to learn residual mappings. Fig 4 shows the InceptionResNetV2 version, combining Inception modules and residual connections for efficient image classification.

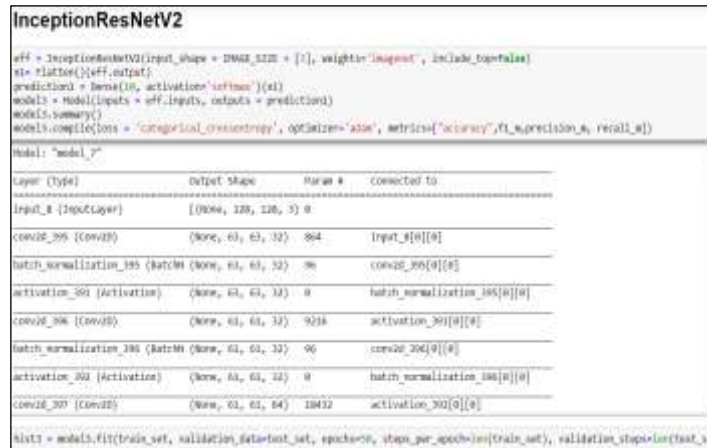


Figure 4: InceptionResnetV2.

Source: Authors, (2026).

**InceptionV3**, in the context of image recognition and class, InceptionV3 is the popular solution. This is a CNN model. This version changed into created by Google and is part of the Inception family. Incorporating factorized convolutions and batch normalization, InceptionV3 improves upon InceptionV2, ensuing in a more efficient and accurate model. Its robust performance in image class obligations concerning massive datasets has brought about its big adoption in computer vision packages. Figure 5 illustrates the InceptionV3 model, which employs factorized convolutions and batch normalization to enhance image classification efficiency.

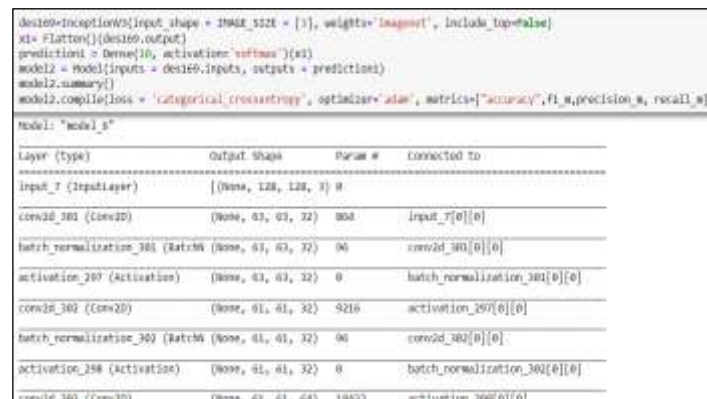


Figure 5: Inception V3.

Source: Authors, (2026).

**ResNet152V2**, The ResNet152V2 family of deep convolutional neural networks represents an enhancement of the ResNet (Residual Networks) structure. The system brings out a 152-layer ResNet which utilizes residual connections to enhance training performance of deep structures. The incorporation of refinements designed to enhance training performance and basic speed, in conjunction with bottleneck blocks and other enhancements, shows that ResNet "V2" represents a development over the unique ResNet. ResNet152V2 is the endorsed preference when a robust deep neural community is necessary for image class and different computer vision duties. Figure 6 illustrates the ResNet152V2 model, a deep neural community that employs residual connections to decorate training efficiency and accuracy.

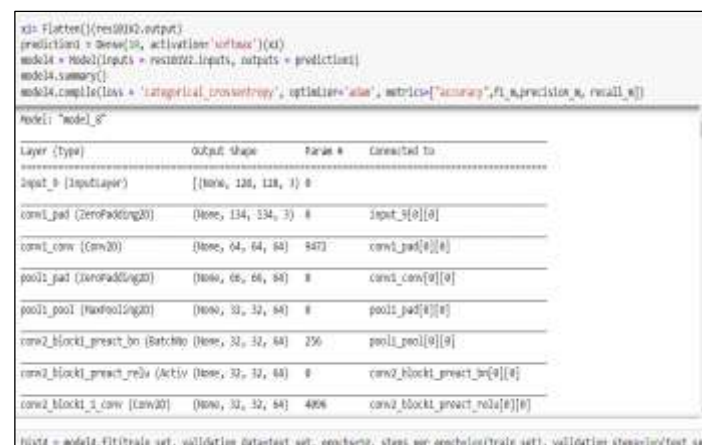


Figure 6: ResNet152V2.

Source: Authors, (2026).

**PCA DeepNet**, The PCA DeepNet architecture integrates traditional ML techniques, consisting of PCA, with a customized DNN. Deep neural networks are educated for infection identity through pattern recognition and classification, while principal component analysis (PCA) serves as the principal function extractor, reducing the dimensionality of input information. A unique amalgamation of classical and DL methodologies ambitions to enhance the performance and accuracy of ailment identification. Figure 7 illustrates the PCA DeepNet model, which integrates major thing analysis with a deep neural community for the purpose of disease categorization.

```
# Input layer
Inputt = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3), name="InputLayer")

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

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

# Transformer Block
encodings = TransformerBlock(tokens)

# Sequence Pooling
pooled = SeqPool(encodings)

# Classifier
Outputt = Dense(n_classes, activation='softmax', name="OutputLayer")(pooled)

# Model
model = Model(Inputt, Outputt, name="CCT")
model.summary()
```

Figure 7: PCA DeepNet.  
Source: Authors, (2026).

**KNN**, K-Nearest Neighbors (KNN) it's far a sincere gadget learning methodology that encompasses each regression and category responsibilities. It considers the average or maximum established category among the k-nearest neighbors of a data factor when designating a label to a new information point. The choice of 'k' extensively affects the performance of the approach; as an instance, while KNN is simple to build, it cannot perform optimally on larger datasets. Definitely. Figure 8 illustrates the k-Nearest neighbors (KNN) algorithm, which categorizes records in line with the fundamental elegance of its nearest neighbors.

```
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)
```

Figure 8: KNN.  
Source: Authors, (2026).

**Decision Tree**, although decision trees are predominantly employed for classification problems, they also can be applied to regression tasks as a supervised learning technique. The classifier is organized as a tree, with dataset properties at the root, decision criteria on the branches, and consequences at the leaf nodes. Decision Node and Leaf Node constitute essential types of nodes within a DT. Figure 9 illustrates the DT model, which categorizes data by partitioning it according to feature values at each 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)
```

Figure 9: Decision tree.  
Source: Authors, (2026).

**Random Forest**, Random forest is a prominent ML algorithm utilized in machine learning. In machine learning, it performs effectively in both type and regression tasks. The core concept is ensemble learning, which integrates multiple classifiers to address a complex problem and improve the model's efficacy. As indicated by way of its nomenclature, "Random forest" is a class approach that consists of numerous decision trees generated from numerous subsets of the provided dataset, which can be aggregated to enhance the predicted accuracy of the dataset. Figure 10 illustrates the "Random forest model, an ensemble of decision trees" employed to decorate classification accuracy.

**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)
```

Figure: 10 Random forest.  
Source: Authors, (2026).

**Gradient Boosting**, The ensemble approach of machine learning, known as Gradient Boosting, constructs a robust predictive version by means of amalgamating the predictions of weak learners, generally decision trees. It rectifies inaccuracies incrementally by means of training supplementary learners on the residual errors of the existing ensemble. The technique allocates a weight to every learner in step with the quantity of effort they utilized in reducing errors. The resistance of Gradient Boosting to overfitting and its exceptional predictive accuracy have made it a preferred choice. Figure 11 illustrates the Gradient Boosting method, which constructs robust classifiers by way of gradually integrating several weak learners.

**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)
```

Figure :11 Gradient boosting.  
Source: Authors, (2026).

**Densenet, considering** The DenseNet design of convolutional neural networks is most desirable for image categorization responsibilities. It promotes characteristic reutilization through its comprehensive interconnectedness, in which all layers attain records from all preceding layers. To attain parameter efficiency, DenseNet employs bottleneck layers for efficiency and transition layers for dimensionality control. This allows it to rival the performance of traditional CNNs even as utilizing fewer parameters. Figure 12 illustrates the DenseNet model, in which every layer is interconnected with all different layers to enhance feature reutilization and performance.

**DenseNet**

```
model = DenseNet201(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
cin = Flatten()(model.output)
prediction = Dense(10, activation='softmax')(cin)
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy', f1_score, precision_score, recall_score])
model.summary()
```

| Layer (type)                             | Output Shape            | Param # | Connected to          |
|------------------------------------------|-------------------------|---------|-----------------------|
| input_2 (InputLayer)                     | (None, 128, 128, 3) 0   |         |                       |
| zero_padding2_1 (ZeroPadding2D)          | (None, 134, 134, 3) 0   |         | input_2[0][0]         |
| conv2d_conv (Conv2D)                     | (None, 64, 64, 64) 9408 |         | zero_padding2_1[0][0] |
| conv2d_bn (BatchNormalisation)           | (None, 64, 64, 64) 256  |         | conv2d_conv[0][0]     |
| conv2d_relu (Activation)                 | (None, 64, 64, 64) 0    |         | conv2d_bn[0][0]       |
| zero_padding2_2 (ZeroPadding2D)          | (None, 66, 66, 64) 0    |         | conv2d_relu[0][0]     |
| pool1 (MaxPooling2D)                     | (None, 32, 32, 64) 0    |         | zero_padding2_2[0][0] |
| conv2d_bottleneck_1 (BatchNormalisation) | (None, 32, 32, 64) 256  |         | pool1[0][0]           |

lists = model.fit(train\_set, validation\_data=test\_set, epochs=10, steps\_per\_epoch=len(train\_set), validation\_steps=len(test\_set))

Figure 12: Densenet.  
Source: Authors, (2026).

**Xception**, Xception is a complicated deep learning framework, representing an excessive iteration of Google's Inception (GoogLeNet) architecture. On this context, "Xception" denotes "extreme Inception." Deepwise separable convolutions, which decompose normal convolution into its critical additives—depthwise and pointwise—supplant conventional Inception modules. The objective of this adjustment is to decrease the range of parameters within the network while improving its capability to identify problematic patterns [8]. Figure 13 illustrates the Xception model, which employs depthwise separable convolutions to enhance performance and diminish complexity.

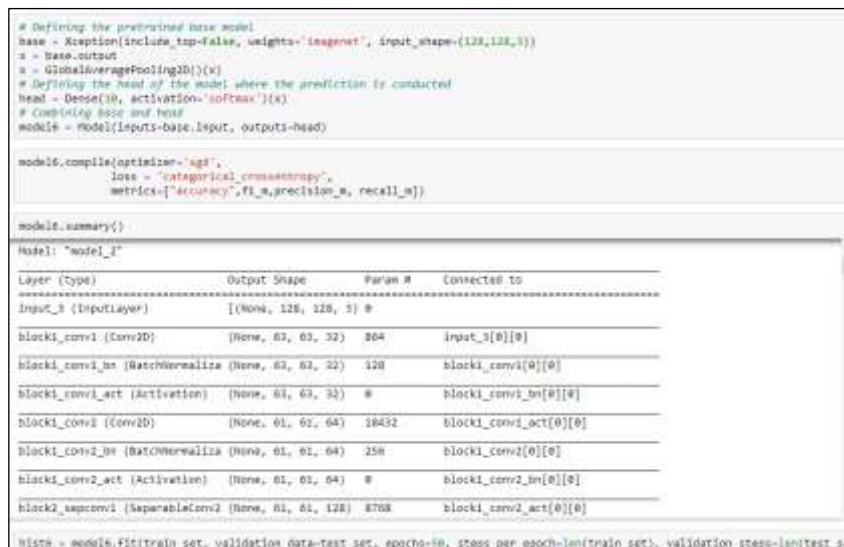


Figure 13: Xception.  
Source: Authors, (2026).

**Voting Classifier**, voting classifiers are machine learning algorithms that examine by means of training on numerous models and expect an output (class) by means of identifying the class with the highest likelihood of being the result. It calculates the common consequences from every classifier inside the voting classifier to determine the output magnificence based totally on the magnificence that gets the largest variety of votes. The objective is to develop a novel model that integrates insights from various specialized fashions and generates predictions based totally on most of the people consensus across all output categories, in preference to constructing and assessing individual specialized models for accuracy. Figure 14 illustrates the voting Classifier, which amalgamates predictions from many models to enhance overall type precision.

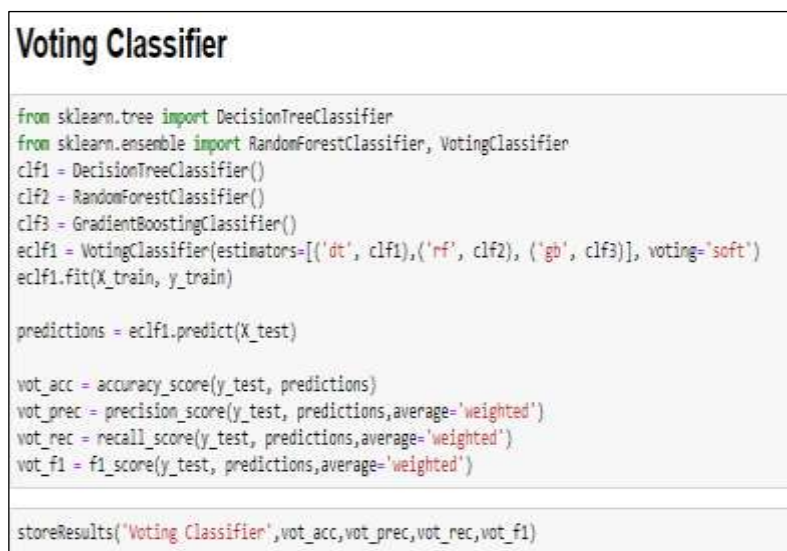


Figure 14: Voting classifier.  
Source: Authors, (2026).

**Faster RCNN**, faster R-CNN is a DL version mainly evolved for item detection duties in computer vision. It consists of a region proposal network (RPN) able to effectively figuring out regions in a photo which can incorporate an item. The prediction of item bounding boxes and class opportunity is predicated on location proposals produced by way of the region proposal network (RPN). The model excels at handling complicated situations involving multiple factors and has established a reputation for accuracy. To precisely hit upon & find diseases in images of tomato plants, have a look at employs faster R-CNN due to its effective -stage structure, which incorporates a area inspiration community, in addition to its established reliability in item detection. It appears you intended to put up a certain paragraph for paraphrasing; however, I do no longer observe any content following the brackets. Figure 15 illustrates the faster R-CNN model, which employs a place inspiration network for particular object detection.

```

# get the model using our helper function
model = get_model_byov(num_classes)

...

# use this to reset all trainable weights
model.apply(reset_weights)

...

# save model to the right device
model.to(device)

# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, # Feel free to play with values
                             momentum=0.9, weight_decay=0)

# Defining learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                                step_size=20,
                                                gamma=0.2)

result_mAP = []
best_epoch = None

# Let's train!
for epoch in range(num_epochs):

    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    results = evaluate(model, data_loader_test, device=device)
    # save results of mAP @ IoU = 0.5
    result_mAP.append(results.coco_eval['bbox'].stats[1])
    # save the best model so far
    if result_mAP[-1] == max(result_mAP):
        best_save_path = os.path.join(f'bestmodel_novug_sqd{wlr=81_batches-epoch[epoch].pth')
        torch.save(model.state_dict(), best_save_path)
        best_epoch = int(epoch)
        print(f'\nmodel from epoch number {epoch} saved! in result is {max(result_mAP)}\n\n')

# Saving the last model
save_path = os.path.join(f'novug_sqd_batches-lastepoch{num_epochs-1}.pth')
torch.save(model.state_dict(), save_path)

```

Figure 15: Faster RCNN.

Source: Authors, (2026).

**YOLOv5 (You Only Look Once version 5):** The YOLOv5 object detection model employs a grid to partition an image into cells, and in a single community pass, it predicts bounding boxes and class probabilities for each cell. It is an advanced version of the YOLO collection that attains a fantastic equilibrium between velocity and accuracy, making it appropriate for duties necessitating real-time object detection. Figure 16 illustrates the YOLOv5 version, which executes real-time object identity through a unique community assessment.

Yolov 5x6

!wandb disabled

python train.py --img 416 --batch 2 --epochs 200 --data /content/drive/MyDrive/7/yolo/data.yaml --weights yolov5x6.pt --cache

Overriding model.yaml nc=40 with nc=100

|    | from    | n  | params   | module                               | arguments             |
|----|---------|----|----------|--------------------------------------|-----------------------|
| 0  | -1      | 1  | 6800     | models.common.Conv                   | [3, 80, 6, 2, 2]      |
| 1  | -1      | 1  | 115520   | models.common.Conv                   | [80, 160, 3, 2]       |
| 2  | -1      | 4  | 309120   | models.common.C3                     | [160, 160, 4]         |
| 3  | -1      | 1  | 461440   | models.common.Conv                   | [160, 320, 3, 2]      |
| 4  | -1      | 8  | 2259200  | models.common.C3                     | [320, 320, 8]         |
| 5  | -1      | 1  | 1844400  | models.common.Conv                   | [320, 640, 3, 2]      |
| 6  | -1      | 12 | 13125120 | models.common.C3                     | [640, 640, 12]        |
| 7  | -1      | 1  | 5531520  | models.common.Conv                   | [640, 960, 3, 2]      |
| 8  | -1      | 4  | 11070720 | models.common.C3                     | [960, 960, 4]         |
| 9  | -1      | 1  | 11061760 | models.common.Conv                   | [960, 1280, 3, 2]     |
| 10 | -1      | 4  | 19676160 | models.common.C3                     | [1280, 1280, 4]       |
| 11 | -1      | 1  | 4095040  | models.common.SPPF                   | [1280, 1280, 5]       |
| 12 | -1      | 1  | 1230720  | models.common.Conv                   | [1280, 960, 1, 1]     |
| 13 | -1      | 1  | 0        | torch.nn.modules.upsampling.Upsample | [None, 2, 'nearest']  |
| 14 | [-1, 8] | 1  | 0        | models.common.Concat                 | [1]                   |
| 15 | -1      | 4  | 11992320 | models.common.C3                     | [1920, 960, 4, False] |
| 16 | -1      | 1  | 615600   | models.common.Conv                   | [960, 640, 1, 1]      |

Figure 16: YOLOV5.

Source: Authors, (2026).

**YOLOv8 (You Only Look Once version 8):** if you are searching for an advanced computer imaginative and prescient model gifted in item identity, segmentation, and posture estimation, choose YOLOv8. Displaying a sophisticated layout and cutting-edge features, it gives amazing performance, advancing the subculture of YOLO. Real-time item tracking, image class, and pose estimation are only some of the myriad packages that benefit from YOLOv8's rapidity, accuracy, and user-friendliness. Figure 17 illustrates the YOLOv8 model, and more advantageous iteration of YOLO engineered for fast and precise object identification.

```

from ultralytics import YOLO

# load a model
# model = YOLO("yolov8n.pt") # build a new model from scratch
model = YOLO("yolov8n.pt") # load a pretrained model (recommended for training)

# use the model
results = model.train(data="/content/drive/MyDrive/7/yolo/data.yaml", epochs=200, imgsz=416) # train the model

# save the model
model.save("yolov8n.pt")

Overriding model.yaml nc=88 with nc=10

  from n  params module  arguments
0  -1 1 1302 ultralytics.nn.modules.conv.Conv [3, 48, 1, 2]
1  -1 1 41664 ultralytics.nn.modules.conv.Conv [48, 96, 1, 2]
2  -1 2 111360 ultralytics.nn.modules.block.C2F [96, 96, 2, True]
3  -1 1 186272 ultralytics.nn.modules.conv.Conv [96, 192, 1, 2]
4  -1 4 82312 ultralytics.nn.modules.block.C2F [192, 192, 4, True]
5  -1 1 66432 ultralytics.nn.modules.conv.Conv [192, 384, 1, 2]
6  -1 4 324944 ultralytics.nn.modules.block.C2F [384, 384, 4, True]
7  -1 1 1991808 ultralytics.nn.modules.conv.Conv [384, 576, 1, 2]
8  -1 2 989504 ultralytics.nn.modules.block.C2F [576, 576, 2, True]
9  -1 1 831168 ultralytics.nn.modules.block.C2F [576, 576, 1]
10 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
11 [-1, 6] 1 0 ultralytics.nn.modules.conv.Concat [1]
12 -1 2 1893728 ultralytics.nn.modules.block.C2F [576, 576, 2]
13 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
14 [-1, 4] 1 0 ultralytics.nn.modules.conv.Concat [1]
15 -1 2 517632 ultralytics.nn.modules.block.C2F [576, 192, 2]

```

Figure 17: YOLOV8.  
Source: Authors, (2026).

#### IV. EXPERIMENTAL RESULTS

**Accuracy:** Correct test results represent the extent to which the test differentiates health cases from affected cases. The accuracy assessment of a test demands the calculation of actual positive cases and actual negative cases across all testing instances. The mathematical expression of this concept becomes:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

**Precision:** The accuracy is computed through the proportion of appropriate classifications among persons identified as positive. Accuracy can be calculated through this formula to evaluate suitable identification among positive cases.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (2)$$

**Recall:** In device learning recall represents a measurement which evaluates how well a model recognizes all necessary cases belonging to a particular class. The recall rate expresses how many correct predictions should exist among all actual positive results to demonstrate how effectively the model understands particular type occurrences.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

**F1-Score:** The F1 rating functions as a metric tool which enables users to assess gadget mastering version accuracy levels. The F1 metric combines the precision along with recall metrics of a system. Accurate predictions from a model can be measured by the accuracy metric which tracks the proper predictions across the entire dataset.

$$\text{F1 Score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} * 100 \quad (1)$$

**mAP:** Mean average Precision (MAP) operates as a statistical measurement to analyze rating contentment. This measurement considers both the number of suitable recommendations and their order within the generated list. The Mean Average Precision (MAP) at ok consists of computing the arithmetic mean of all Average Precision (AP) at k values for each customer query.

$$\text{mAP} = \frac{1}{n} \sum_{k=1}^n AP_k \quad (5)$$

Table 1: Performance Evaluation Table – Classification.

| 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      |

Source: Authors, (2026).

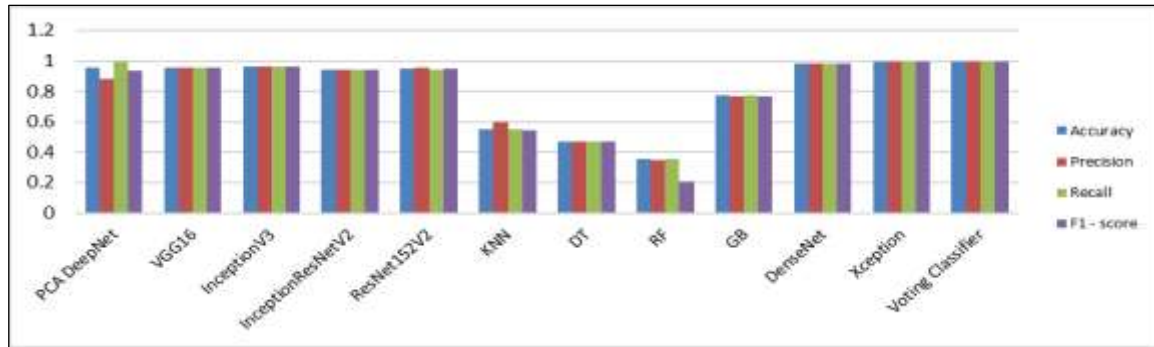
**Table (1)** presents a comparison of various ML models based on Accuracy, Precision, Recall, and F1-score. The **Voting Classifier** outperforms all other algorithms, achieving the highest performance across all metrics.

Table 2: Performance Evaluation Table – Detection.

| Algorithms  | Precision | Recall | mAP  |
|-------------|-----------|--------|------|
| YoloV5      | 0.977     | 0.98   | 0.99 |
| YoloV8      | 0.984     | 0.98   | 0.99 |
| Faster RCNN | 0.698     | 0.75   | 0.75 |

Source: Authors, (2026).

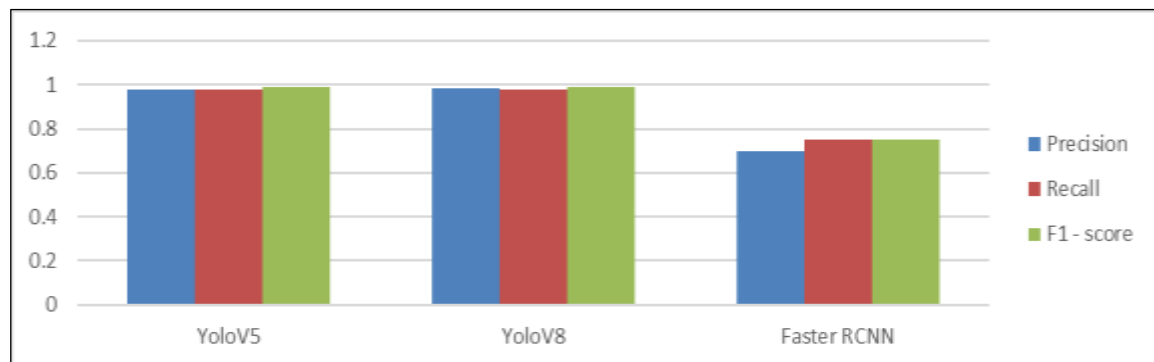
**Table (2)** presents a comparison of various DL models based on Precision, Recall, and mAP. The YoloV8 outperforms all other algorithms, achieving the highest performance.



Graph 1: Comparison Graph – Classification.

Source: Authors, (2026).

**Graph (1)** compares various ML models based on four evaluation metrics. Accuracy is represented in blue, Precision in red, Recall in light green, and F1-score in violet. The **Voting Classifier** outperforms all models, achieving the highest performance across all metrics.



Graph.2: Comparison Graph-Detection.

Source: Authors, (2026).

**Graph (2)** compares various deep learning models based on three evaluation metrics. Precision in blue, Recall in red, and mAP in light green. The YoloV8 outperforms all models, achieving the highest performance across all metrics.

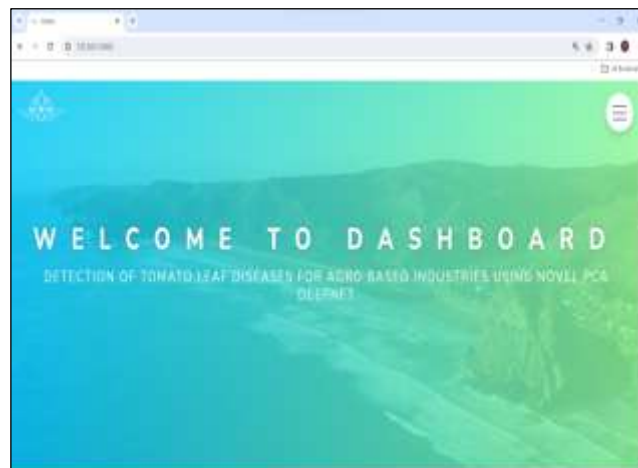
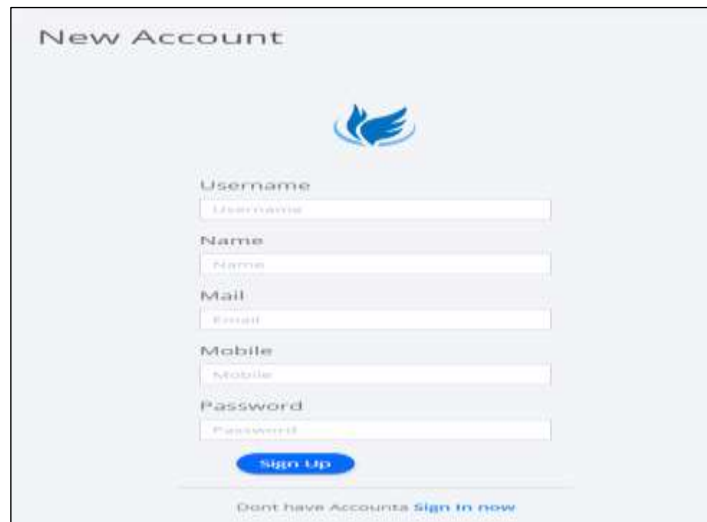


Figure 18: Home page.

Source: Authors, (2026).

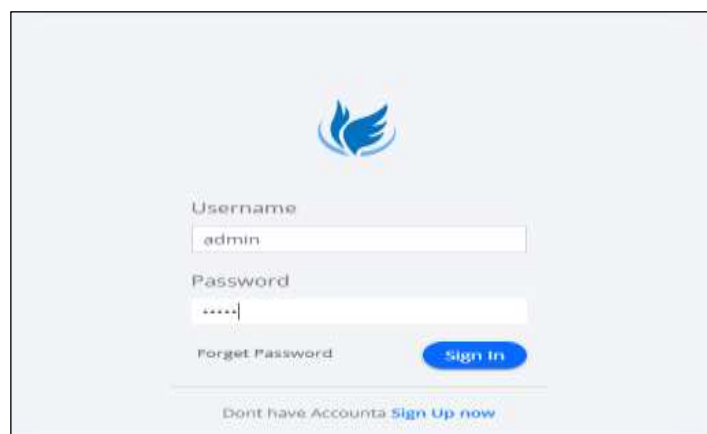
The homepage has a well-designed dashboard whose background is a gradient and it greets the users to the tomato leaf disease detection system of PCA DeepNet.



The registration page features a light blue background with a stylized blue flame logo at the top center. Below the logo, the title 'New Account' is displayed. The form includes five input fields: 'Username', 'Name', 'Mail', 'Mobile', and 'Password'. Each field has a placeholder text corresponding to its label. A blue 'Sign Up' button is positioned below the 'Password' field. At the bottom, there is a link that says 'Don't have Accounts Sign in now'.

Figure 19: Registration page.  
Source: Authors, (2026).

Registration page enables users to develop a new account by providing the following details: username, name, email, mobile number and password.



The login page has a light blue background with the same stylized blue flame logo at the top center. Below the logo, the title 'Login' is displayed. The form includes two input fields: 'Username' and 'Password'. The 'Username' field contains the text 'admin'. The 'Password' field contains six dots. A blue 'Sign In' button is located to the right of the 'Password' field. Below the 'Sign In' button, there is a link that says 'Don't have Accounts Sign Up now'.

Figure 20: Login page.  
Source: Authors, (2026).

The log in page enables one to log in with the help of username and password, along with password recovery and account creation possibilities.

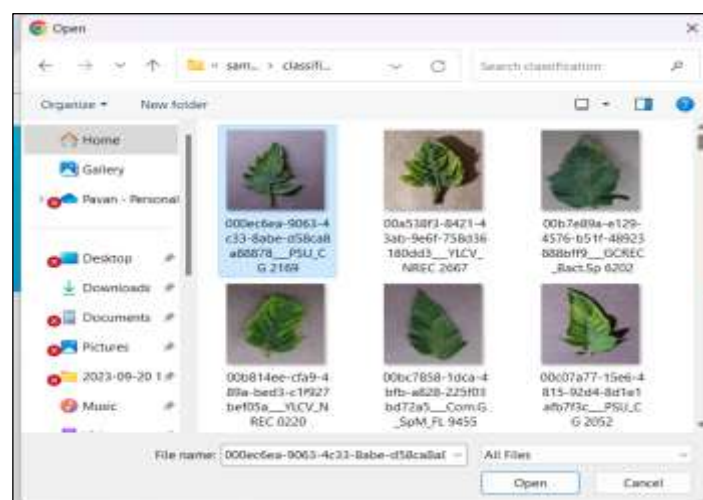


Figure 21: Upload input image – Classification.  
Source: Authors, (2026).

The upload input image classification page is where the user can choose and access the leaf images to classify them and help in the identification of the disease.



Figure 22: Predicted Results.  
Source: Authors, (2026).

The result page of categorizing leaves provides the provided leaf image together with the projected disorder, categorizing it as inflamed with the Tomato Mosaic Virus. Figure 22 represents the type output, which gives the sent image of the tomato leaf with the projected analysis of illnesses.

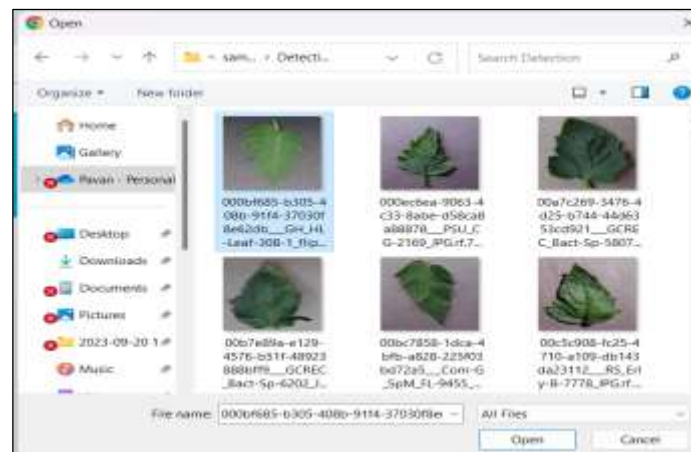


Figure.23: Upload Input Image – Detection.  
Source: Authors, (2026).

As shown in the detection page, the user is presented with a file upload dialog where he/she chooses a leaf image to be analysed to identify possible diseases. Fig. 23 illustrates the detection input page, on which users will provide a leaf image on which they want to analyze to detect the disease.



Figure 24: Predict result considering given input.  
Source: Authors, (2026).

The picture depicts a healthy green leaf with a confidence of 96% which is marked with an orange bounding box in a dark background.

## V. CONCLUSIONS

The research aims to provide the full solution to the specific disorder identification in tomato leaves through a combination of deep learning models (PCA DeepNet, VGG16, InceptionResNetV2) with conventional ML algorithms (KNN, DT, RF) [6]. The model is then aimed at rendering the model sensitized to different agricultural environments by careful image processing, including rescaling and editing. In order to provide correct disease detection, purposeful version architecture, in combination with VGG16 and InceptionResNetV2, uses a strategic approach to record global and specific statistics. The first-rate performance of the models has been proven by various methods, showing that large ensemble contemporary models are of high quality in terms of type, whereas the state of the and YoloV8 models are detection models with high accuracy. Its accuracy and viability in determining and locating tomato leaf diseases has been also demonstrated by mammoth checking out in an accessible the front end the use of actual feature values [4] [6] [10]. The current Flask and SQLite support the seamless integration that allows farmers to check in, add pictures, and get hold of activate analyses so that they can make informed decisions. Which makes it easier to apply the modern precision agriculture to the use of state AI to detect disease in time. This is a proactive strategy, which maintains crop satisfactory and promotes sustainable agricultural practices.

## VI. FUTURE SCOPE

In the agricultural sector, farmers and other players in the industry can achieve on the spot advantages by automating the processes into a real time machine that detects and averts diseases early in relation to tomato leaves [4]. This research was predominantly based on the evaluation of the suggested PCA DeepNet classifier in comparison with other existing deep learning models and the hyper parameters used. We can find additional optimization of the system. By adding additional agricultural platforms and technology to the system, it can be improved to be more accessible and user-friendly. This can be used to enhance assimilation into other agricultural practices. The system can be enhanced to identify and categorize diseases in different crops [28-36] to complement its software and impact in the agricultural process.

## VII. AUTHOR'S CONTRIBUTION

**Conceptualization:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Methodology:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Investigation:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Discussion of results:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Writing – Original Draft:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Writing – Review and Editing:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Resources:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Supervision:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

**Approval of the final text:** Yuvaraj Singh Tagore, Manna Sheela Rani Chetty.

## VIII. REFERENCES

- [1] Y. Wu, X. Feng, & G. Chen, "Plant leaf diseases fine-grained categorization using convolutional neural networks," *IEEE Access*, vol. 10, pp. 41087–41096, 2022.
- [2] M. Adnan, K. Ali, G. Drushti, & 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, & 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, & 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, & I. Ahmad, "An efficient approach considering crops pests recognition & 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, & M. H. Kabir, "Less is more: Lighter & faster deep neural architecture considering tomato leaf disease classification," *IEEE Access*, vol. 10, pp. 68868–68884, 2022.
- [7] E. Elfatimi, R. Eryigit, & 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, & S. Rondinella, "Tomato diseases classification based on VGG & 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, & 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, & 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, & M. S. Islam, "Disease detection of plant leaf using image processing & 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, & M. Mahar, "Plant disease detection using deep learning," *Int. J. Recent Technol. Eng.*, vol. 9, no. 1, pp. 909–914, 2020.

- [13] A. Rao & S. B. Kulkarni, “A hybrid approach considering plant leaf disease detection & 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, & M. Singh, “Detection & 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 & B. Joshi, “A survey on plant leaf disease detection techniques,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 6, no. 1, pp. 229–231, 2017.
- [16] J. Gui, L. Hao, Q. Zhang, & X. Bao, “A new method considering 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, & 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 & N. C. Htun, “Plant leaf disease detection & 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, & A. M. Rad, “An investigation into machine learning regression techniques considering 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, & D. S. Park, “Unsupervised image translation using adversarial networks considering improved plant disease recognition,” *Comput. Electron. Agricult.*, vol. 168, Jan. 2020, Art. no. 105117.
- [21] Q. Zeng, X. Ma, B. Cheng, E. Zhou, & W. Pang, “GANs-based data augmentation considering 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, & 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, & B. Wang, “Plant disease detection & classification by deep learning—A review,” *IEEE Access*, vol. 9, pp. 56683–56698, 2021.
- [24] M. Brahimi, S. Mahmoudi, K. Boukhalfa, & A. Moussaoui, “Deep interpretable architecture considering plant diseases classification,” in *Proc. Signal Process., Algorithms, Architectures, Arrangements, Appl. (SPA)*, Sep. 2019, pp. 111–116.
- [25] J. G. A. Barbedo, “Factors influencing use of deep learning considering 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, & 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, & 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, & K. A. Sada, “Plant disease detection with deep learning & feature 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, & M. Imran, “A deep learning-based framework considering automatic brain tumors classification using transfer learning,” *Circuits, Syst., Signal Process.*, vol. 39, no. 2, pp. 757–775, Feb. 2020.
- [31] Y. H. Bhosale & K. Sridhar Patnaik, “IoT deployable lightweight deep learning application considering 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, & S. K. Rath, “Classification of sentiment reviews using n-gram machine learning approach,” *Expert Syst. Appl.*, vol. 57, pp. 117–126, Sep. 2016.
- [33] V. V. Srinidhi, A. Sahay, & K. Deeba, “Plant pathology disease detection in apple leaves using deep convolutional neural networks: Apple leaves disease detection using EfficientNet & 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 & 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, & C. Liang, “Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks,” *IEEE Access*, vol. 7, pp. 59069–59080, 2019.