CHAPTER – V

RESULTS AND ANALYSIS

5.1 Introduction: An Overview of the Chapter

This section provides a thorough analysis of the results obtained from the experiments carried out in this study. This chapter is organized to give a thorough assessment of the effectiveness of the different models used for weed and crop categorization and density estimate. The analysis provides a comprehensive and lucid overview of the model's efficacy and correctness by incorporating a variety of statistical measures and visual aids.

The chapter begins with a presentation of the descriptive statistics, summarizing the key characteristics of the dataset used in the experiments. This section sets the foundation for understanding the data distribution and its implications for model performance.

Following the descriptive statistics, the results of the classification models are detailed. A number of performance metrics are used to assess each model, including the Customized CNN from Scratch, Model with Image Augmentation, Transfer Learning with VGGNet, and Transfer Learning with ResNet50. These metrics include accuracy, precision, recall, and F1-score. In order to shed light on the advantages and disadvantages of each model and provide comparisons regarding their respective performances, comparative assessments are carried out.

The chapter then delves into the results of the YOLOv8 model for crop and weed density estimation. This section includes an analysis of the model's detection accuracy, processing speed, and its applicability to real-time agricultural monitoring. The model's predictions are shown through visual examples and heatmaps, which provide a concrete understanding of how well it performs in real-world situations.

Subsequently, the chapter discusses the inferential statistics, focusing on hypothesis testing results and confidence intervals. This section interprets the statistical significance of the findings, linking them back to the research questions and hypotheses outlined in the earlier chapters.

A thorough analysis of the findings and their implications for precision agriculture are included in this chapter. The results are discussed in relation to agricultural methods, technology developments, and the possibility of further research. The last section of

the chapter sets the stage for the thesis's last chapter by summarizing the major discoveries and their contributions to the discipline.

5.2 Model Architecture Selection

Any machine learning activity, including agricultural categorization, depends on choosing a suitable model architecture. This section covers the process of choosing a model architecture and lists the four models that were taken into consideration for our agricultural classification task: a VGGNET and ResNet50 architecture-based transfer learning approach, an augmented version of the customized CNN, and a customized CNN built from scratch.

5.2.1 Model-1: Customized CNN from Scratch

The first model we consider is a customized CNN architecture built from scratch specifically for the agricultural classification task. This model includes several convolutional layers for feature extraction and spatial dimension reduction, which are followed by max-pooling layers. Subsequently, fully connected layers are employed for classification, with softmax activation at the output layer to generate class probabilities.

We can customize the architecture of a CNN to the unique features of the agricultural photos and the difficulty of the classification task by creating a CNN from the ground up. We may finely tune the number of layers, the size of filters, and the connectivity patterns by starting from scratch when constructing the architecture. This allows us to experiment and observe real-world data to maximize the model's performance.

Explanation of the components used in the architecture of model-1

Convolutional Layers (Conv2D):

- The model is comprised of several convolutional layers at first. The learning of the spatial hierarchies of features in the input images is done by these layers.
- The Conv2D layers convolve over the input image using a predetermined number of filters or kernels.
- After every convolution process, activation functions (relu) are added to the model to introduce non-linearity and allow it to learn intricate patterns.
- The first Conv2D layer specifies the input shape of the images and applies a kernel size of 3x3.

- To capture more abstract elements, Conv2D layers after this one add extra filters.

Batch Normalization:

- To normalize the activations of the preceding layer, batch normalization is used after a few convolutional layers. It facilitates and quickens the training process.

MaxPooling2D:

- The input feature maps' spatial dimensions are down sampled using max pooling layers, which lowers computational cost and manages overfitting.

Dropout:

- In order to avoid overfitting, dropout layers randomly deactivate a portion of neurons during training. In this model, the designated dropout rate is 0.25.

Flattening:

- The feature maps are flattened into a one-dimensional vector to be fed into the fully connected layers after the convolutional layers.

Dense (Fully Connected) Layers:

- Dense layers are used for classification. They take the flattened feature vector as input and perform classification based on learned features.
- Activation functions (relu) introduce non-linearity.
- The neurons in the final dense layer are called num_classes, and num_classes is the number of output classes. It outputs class probabilities using a softmax activation function.

Model Compilation:

- The Adam optimizer, which adjusts the learning rate during training, is used to build the model.
- Since categorical cross-entropy is appropriate for multi-class classification issues, it is utilized as the loss function.
- Selecting accuracy as the evaluation metric.

Training:

- The fit function, which provides both training and validation data, is used to train the model.

- The training process is monitored and managed by callbacks over a predetermined number of epochs (epochs=10).

Model evaluation

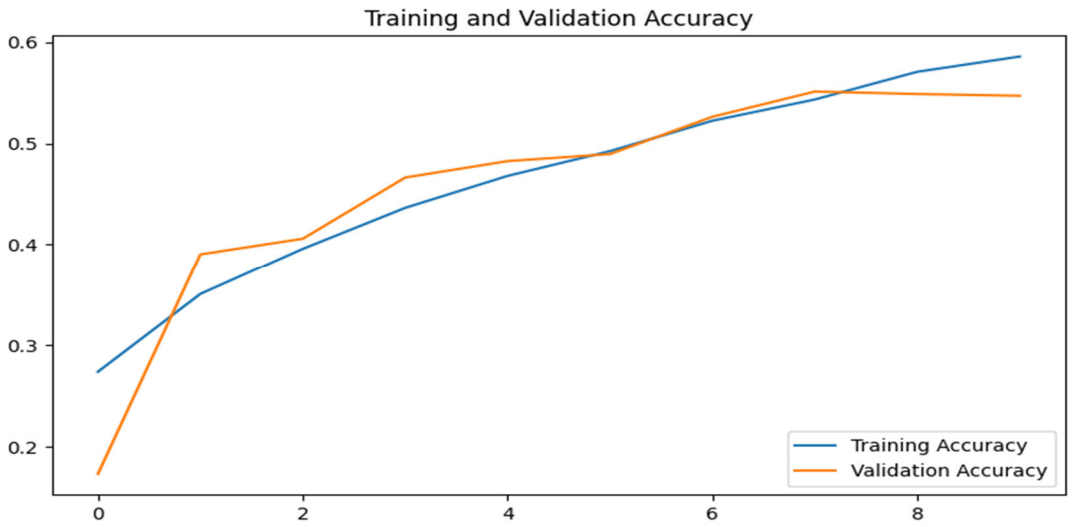

Fig. 5.1 : Model-1 Training Accuracy and Validation Accuracy

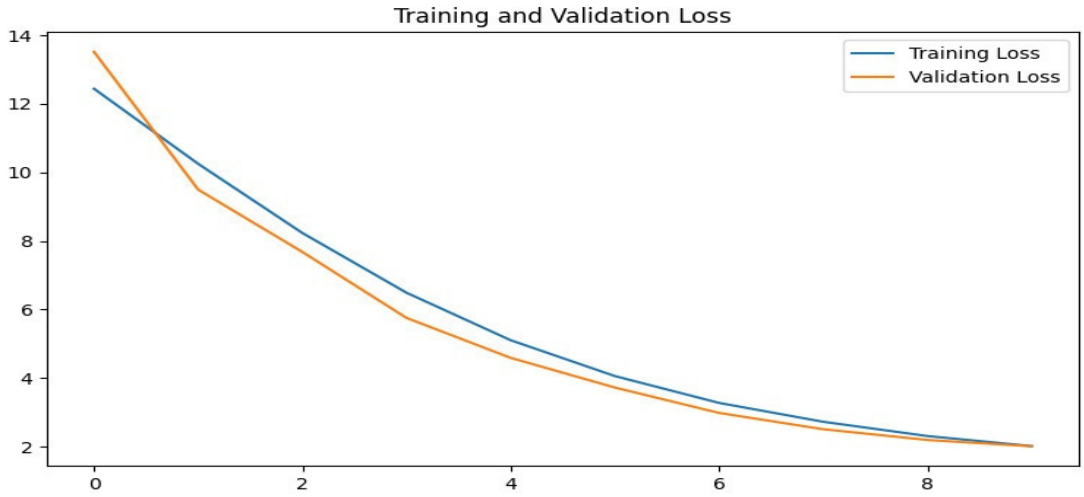

Fig. 5.2 : Model-1 Training Loss and Validation Loss

The performance of our trained model on the training and validation datasets is shown by the model evaluation findings.

Training Dataset Evaluation:

Loss: 1.8045 is the loss value on the training dataset. The discrepancy between the actual labels and the anticipated probabilities is represented by the loss. Better performance is shown by lower loss values.

Accuracy: 62.11% is the accuracy on the training dataset. The percentage of correctly categorized samples in the training dataset is known as accuracy. Better performance is indicated by higher accuracy values.

Validation Dataset Evaluation:

Loss: The validation dataset's loss value is 2.0106. The difference between the true labels and the predicted probabilities on the validation dataset is represented by this loss value.

Accuracy: 55.52% of the validation dataset's data were accurate. The accuracy of the classification indicates the percentage of samples in the validation dataset that were properly classified out of all the samples.

Further analysis is required to understand why the validation accuracy is lower than the training accuracy. Increasing the quantity of training data, fine-tuning hyperparameters, or changing the model architecture are some possible courses of action.

Additionally, monitoring the model's performance over more epochs or using techniques like early stopping will help prevent overfitting and improve generalization to unseen data.

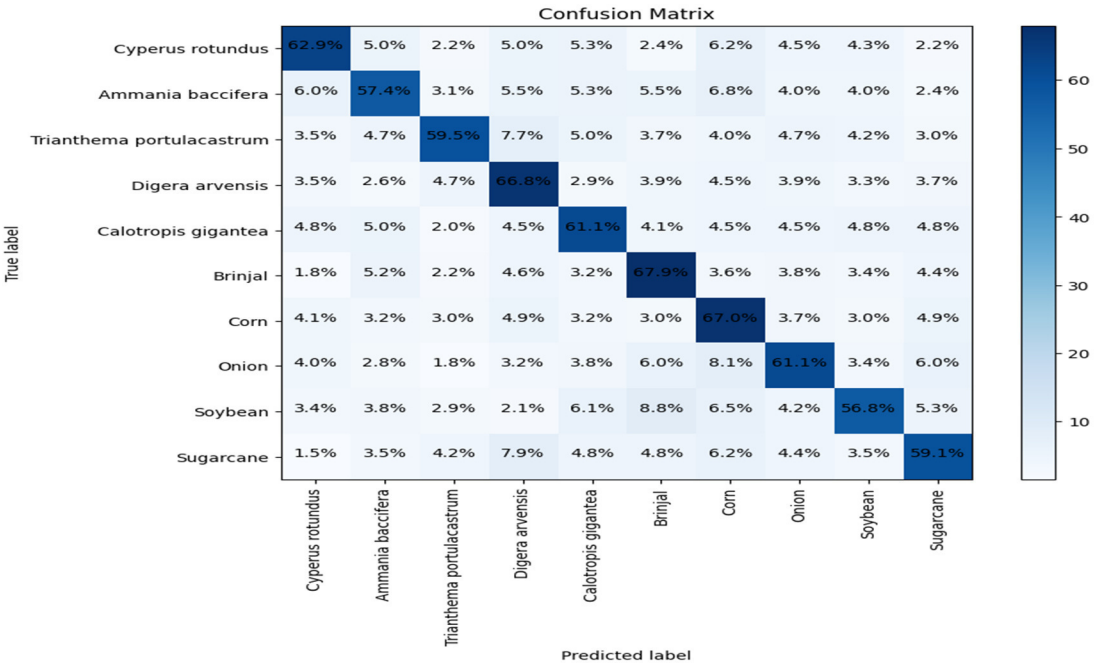

Fig. 5.3 : Model-1 Confusion Matrix

Classification Report of Model-1

	precision	recall	f1-score	support
Cyperus rotundus	0.63	0.63	0.63	418
Ammania baccifera	0.61	0.57	0.59	453
Trianthema portulacastrum	0.65	0.59	0.62	402
Digera arvensis	0.62	0.67	0.64	509
Calotropis gigantea	0.59	0.61	0.60	442
Brinjal	0.63	0.68	0.65	502
Corn	0.62	0.67	0.64	567
Onion	0.64	0.61	0.62	504
Soybean	0.63	0.57	0.60	475
Sugarcane	0.60	0.59	0.60	455
accuracy			0.62	4727
macro avg	0.62	0.62	0.62	4727
weighted avg	0.62	0.62	0.62	4727

Fig. 5.4 : Classification Report of Model-1

5.2.2 Model-2: Model 1 with Image Augmentation

We improve upon Model-1's performance in the second model by using picture augmentation techniques in both the training and validation stages. To increase the diversity of the training dataset, image augmentation entails applying a range of transformations, including rotation, flipping, scaling, and translation, to the input images.

Our goal is to increase the model's capacity for generalization and resilience to changes in the input images by adding more data to the training set. In order to assist the model acquire additional discriminative features and lower the likelihood of overfitting to the training dataset, augmented training data exposes it to a wider range of scenarios and variations.

To build Model-2, we employed the same architecture as Model-1 but introduced image augmentation techniques using the Keras Image Data Generator class.

ImageDataGenerator Configuration: You configured the ImageDataGenerator class with various augmentation options:

Rescaling: Pixel values are rescaled from the range [0, 255] to [0, 1].

Rotation Range: The training images are subjected to random rotation within a predefined degree range, which enhances the model's resistance to changes in object orientation and provides diversity.

Width and Height Shift: Randomly shifts the width and height of the images, providing the model with additional positional information and enabling it to learn from variations in object position within the image.

Shear Range: Introduces shearing transformations to the images, which helps the model learn from distorted perspectives of objects.

Zoom Range: Randomly zooms into or out of the images, enabling the model to learn from variations in scale.

Horizontal Flip: Randomly flips images horizontally, which increases the diversity of training data by presenting mirrored versions of objects.

Fill Mode: Determines the strategy to fill newly created pixels, ensuring that the transformations do not introduce artifacts into the image.

Validation Split: divides the dataset into sets for training and validation so that the model may be assessed while being trained.

Data Flow: For the training, validation, and testing sets, photos and the accompanying classes are automatically retrieved using the flow_from_directory method. During training, images are loaded from the designated directory, scaled to a standard size, and fed into the model in batches.

Model Training: The augmented data produced by the ImageDataGenerator is used to train the model. The model gains knowledge from a variety of augmented images after each training epoch, which helps it perform better and generalize to new data.

Model Evaluation: Using the testing dataset, the model is assessed post-training to determine how well it performs on unobserved data. To outperform Model-1 in terms of resilience and generalization, Model-2 makes use of picture augmentation in both training and evaluation.

Documentation Insights: Recording how image augmentation methods are incorporated into the model-building process shows that improving model

performance and robustness is a proactive approach. It draws attention to the initiatives taken to mitigate any overfitting and enhance the model's capacity to manage the complexities and variances seen in real-world data. It also demonstrates the iterative nature of model creation, where testing various approaches results in incremental gains in the dependability and performance of the model.

Considering the augmentation techniques used, the total number of augmented images generated per original image is:

10 (shear) + 10 (zoom) + 10 (width shift) + 10 (height shift) + 10 (rotation) + 1 (horizontal flip) = 51

As a result, roughly 51 augmented images are produced for every original image, greatly increasing the total number of images that are available for training and validation.

Confusion Matrix:

A thorough analysis of the model's predictions in relation to the real labels can be found in the confusion matrix. The anticipated class is represented by each column, and the actual class is represented by each row.

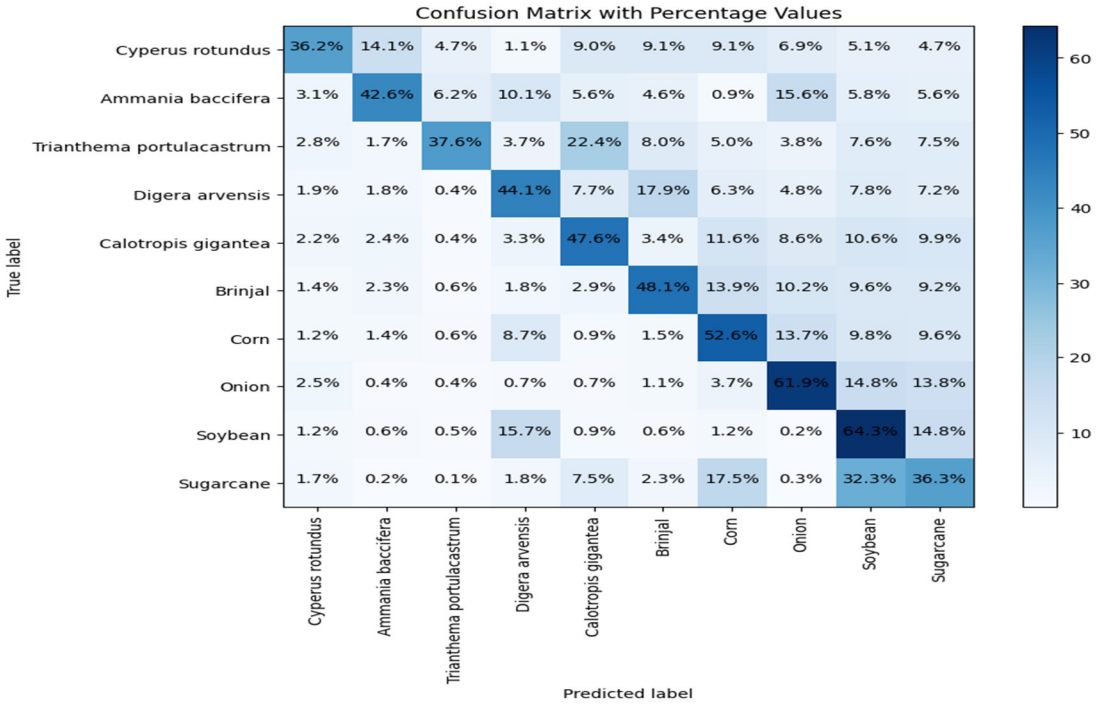

Fig. 5.5 : Model-2 Confusion Matrix

Classification Report:

Along with accuracy, weighted average, macro-average, and precision, recall, and F1-score for each class, the classification report also gives these metrics.

Precision: Shows the percentage of actual positive predictions among all the positive predictions the model made for a given class.

Recall: Calculates the percentage of real positives for a given class that are true positive forecasts.

The F1-score is a balanced indicator of a model's performance that is calculated as the harmonic mean of precision and recall.

Support: The real number of each class's instances in the dataset.

Classification Report of Model-2

	precision	recall	f1-score	support
Cyperus rotundus	0.69	0.36	0.48	1065
Ammania baccifera	0.62	0.43	0.50	970
Trianthema portulacastrum	0.71	0.38	0.49	898
Digera arvensis	0.54	0.44	0.49	1135
Calotropis gigantea	0.43	0.48	0.45	909
Brinjal	0.51	0.48	0.49	1083
Corn	0.46	0.53	0.49	1146
Onion	0.40	0.62	0.49	725
Soybean	0.29	0.64	0.40	661
Sugarcane	0.36	0.36	0.36	1158
accuracy			0.46	9750
macro avg	0.50	0.47	0.46	9750
weighted avg	0.51	0.46	0.46	9750

Fig. 5.6 : Classification Report of Model-2

Model Evaluation:

Accuracy: The model's overall accuracy is 46%, meaning that 46% of the cases in all classes were accurately predicted by the model.

Class-wise Evaluation: We may see differences in performance between classes by examining the precision, recall, and F1-score for each class. For instance, classes like "Onion" and "Soybean" have relatively higher precision and recall compared to classes like "Cyperus rotundus" and "Ammania baccifera."

Macro and Weighted Averages: An overall evaluation of the model's performance across all classes is given by the weighted-average and macro-average scores. In this case, both averages indicate moderate performance, with macro-average being slightly lower due to equal weighting of all classes.

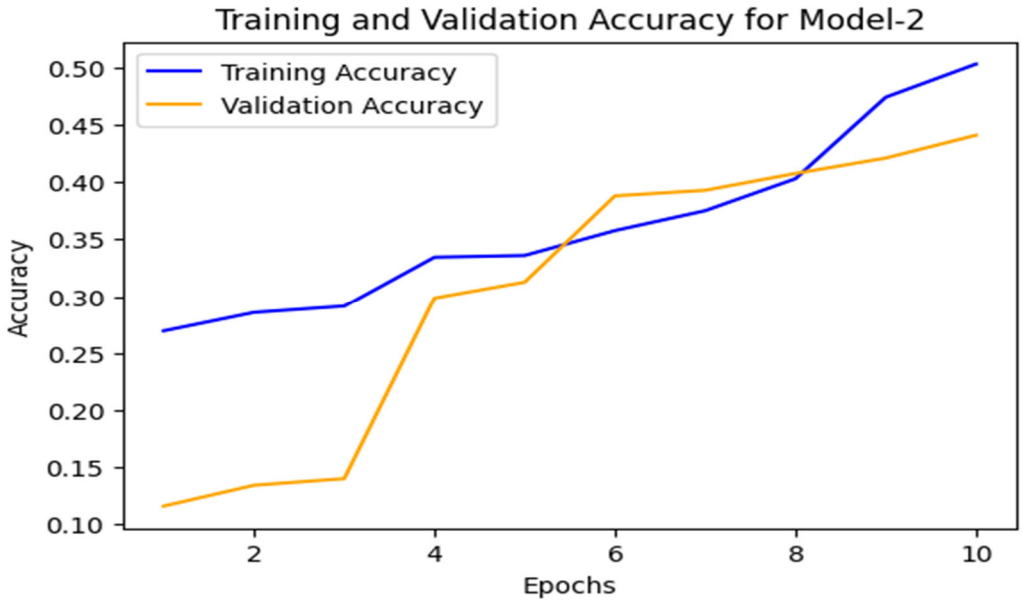

Fig. 5.7 : Model-2 Training Accuracy and Validation Accuracy

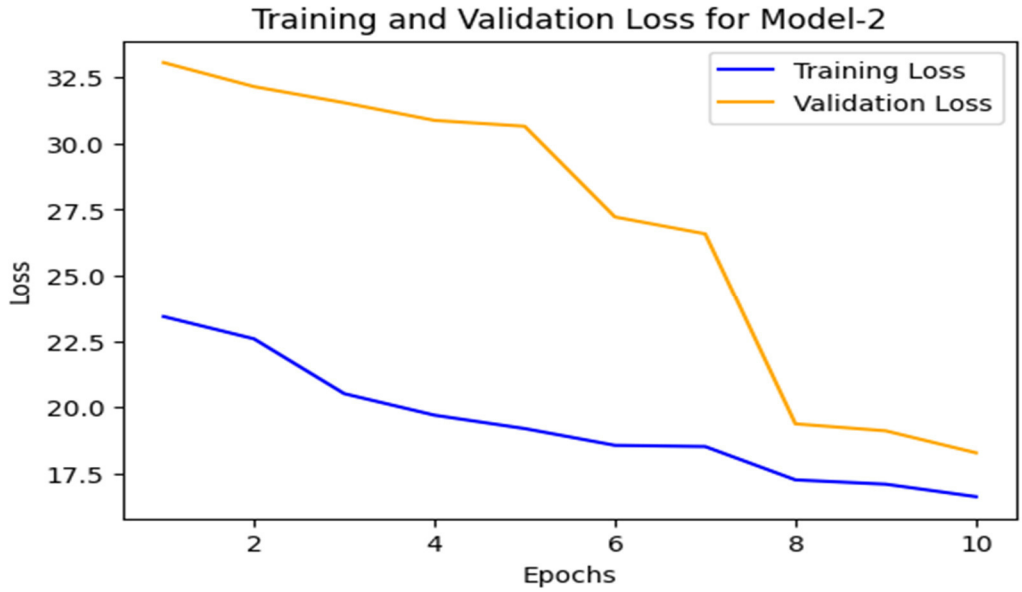

Fig. 5.8 : Model-2 Training Loss and Validation Loss

The model's evaluation suggests that it performs moderately across different classes, with some classes showing better performance than others. The lower precision, recall, and F1-score for certain classes may indicate challenges in accurately

predicting those classes, which could be due to class imbalance, data quality issues, or inherent complexity in distinguishing those classes. Although the model performs better than random guessing, its total accuracy of 46% suggests that more optimization may be needed to increase accuracy and robustness.

5.2.3 Model-3: Transfer Learning with VGGNET

The third model makes use of transfer learning, which is the process of using a previously trained model as a basis for training on a new problem. In particular, we use the base model, which is the VGGNET architecture that has been pre-trained on ImageNet, and refine it using our dataset for agriculture categorization.

Agricultural classification problems can greatly benefit from the rich feature representations learnt from a large-scale dataset such as ImageNet, which can be used through transfer learning using VGGNET. We can tailor the learnt features to the unique properties of agricultural photos by fine-tuning the pre-trained VGGNET model on our dataset. This could result in improved performance with fewer training data and computer resources.

3.4.3.1 VGG16 Architecture Overview

The University of Oxford's Visual Geometry Group proposed the convolutional neural network (CNN) model known as the VGG16 (Visual Geometry Group 16) architecture. It became well-known for being easy to use and efficient when classifying images. An extensive synopsis of the VGG16 architecture is provided below:

1. Convolutional Layers:

   The VGG16 model has thirteen convolutional layers, with a rectified linear unit (ReLU) activation function inserted after each layer to add non-linearity. To preserve the spatial dimensions of the input, the convolutional layers employ tiny receptive fields (3x3) with a stride of 1 and zero-padding.

   The number of filters increases with depth, starting from 64 filters in the first convolutional layer and doubling after each max-pooling layer.

2. Max Pooling Layers:

   Max-pooling layers with a 2x2 filter and a stride of 2 are applied after every two convolutional layers.

By lowering the feature maps' spatial dimensions, max-pooling contributes to a reduction in feature map size and computational complexity.

3. Fully Connected Layers:

   Three fully connected layers, each with 4096 units, come after the convolutional layers.

   With the exception of the output layer, every completely connected layer is followed by a ReLU activation function.

4. Output Layer:

   The last layer is a fully connected layer that represents the probabilities of belonging to each class with 1000 units (for the original ImageNet dataset).

   The raw scores are translated into class probabilities using a softmax activation function.

5. Architecture Summary:

   Input images are typically resized to 224x224 pixels, which is the required input size for VGG16.

   Back propagation along with stochastic gradient descent or other optimization methods is used to train the model end-to-end.

   VGG16's deep architecture and abundance of parameters allow it to perform remarkably well on picture classification tasks.

6. Pre-Trained Model:

   Pre-trained versions of VGG16 are available, trained on large datasets such as ImageNet, which contain millions of labeled images.

   Using the learnt features, transfer learning is often implemented by optimizing these pre-trained models on certain tasks or datasets.

7. Limitations:

   Due to its very large number of parameters, VGG16 requires a lot of memory and processing power.

   Its depth and intricacy may cause overfitting when trained on little datasets.

   In the field of deep learning, VGG16 is a fundamental architecture that acts as a standard for CNN-based image classification models. It is a popular choice for many computer vision tasks due to its modular design and simplicity, but

more contemporary architectures such as ResNet and EfficientNet have outperformed it in terms of efficiency and performance.

Fine-Tuning and Customization

Load VGG16 Base Model: Loads the VGG16 model that has already been trained, excluding its topmost (completely linked) layers. Only the convolutional base is loaded thanks to the include_top=False option.

Make Specified Layers Non-Trainable: To stop the pre-trained layers' weights from changing during training, the weights are frozen up to a certain number (in this case, three layers). In transfer learning, this method is frequently employed to make use of the pre-trained weights and fine-tune only the upper layers for the current job.

Add a Flatten layer to transform the output of the convolutional base into a one-dimensional vector, then add Custom Dense Layers on top of that. Adds custom Dense layers with dropout regularization and ReLU activation algorithms after that. These layers serve as the classification's new completely connected layers.

Output Layer: To estimate the probabilities for each class, add the output layer with softmax activation. The number of classes in your custom classification task is the same as the number of units in this layer.

Establish the Model: specifies the inputs (VGG16 input) and outputs (output layer) to create the custom model.

Put the Model Together: builds the model using the Adam optimizer with a customized learning rate, the categorical cross-entropy loss function, and accuracy as the evaluation metric.

Model Summary: Provides an overview of the layers, output forms, and parameter count in the full model architecture.

This adapts the fully connected layers to the new dataset and efficiently fine-tunes the VGG16 model for your unique classification task. By customizing the model to your unique requirements, fine-tuning enables you to take advantage of the pre-trained weights from the ImageNet dataset. This can result in faster convergence and better performance than if you were to train the model from scratch.

Fine-Tuned and Customized Architecture of Model-3

Training Process

Fitting our customized VGG16 model to our training data and testing it on our test data include training the model.

train_generator: This is our training data generator, which generates batches of training samples and their corresponding labels. It provides data augmentation and preprocessing on-the-fly during training.

steps_per_epoch: During training, the number of steps (batches) to count as one epoch. Usually, the batch size divided by the total number of training samples is used to determine this value.

epochs: The quantity of training dataset iterations (epochs) used to train the model. 50 epocs were utilized to train our model.

Our validation data generator, validation_data, produces batches of validation samples together with the labels that go with them. It is employed to assess how well the model performs during training on a different dataset.

validation_steps: The number of steps (batches) to consider as one evaluation epoch during validation. This value is typically set to the total number of validation samples divided by the batch size.

class_weight: A dictionary that maps class indices to a weight value that is optional. This can be useful for handling class imbalance by giving more weight to minority classes during training.

callbacks: List of callbacks to apply during training. Callbacks are functions that are called at certain points during training (e.g., at the end of each epoch) and can perform actions such as saving the model, adjusting the learning rate, or stopping training early based on certain conditions.

After each epoch, the model is validated using the validation data, and the model is trained on the training data for the predetermined number of epochs using the model.fit() function. The model's weights are updated throughout the training phase in order to minimize the given loss function, in this example, categorical cross-entropy, and its performance is assessed using the given metrics (accuracy).

The loss and accuracy values on the training and validation datasets for each epoch are included in the history object that model.fit() returns. This object provides information about the training process. This information can be used to visualize the training progress and diagnose any issues with overfitting or underfitting.

Model Evaluation

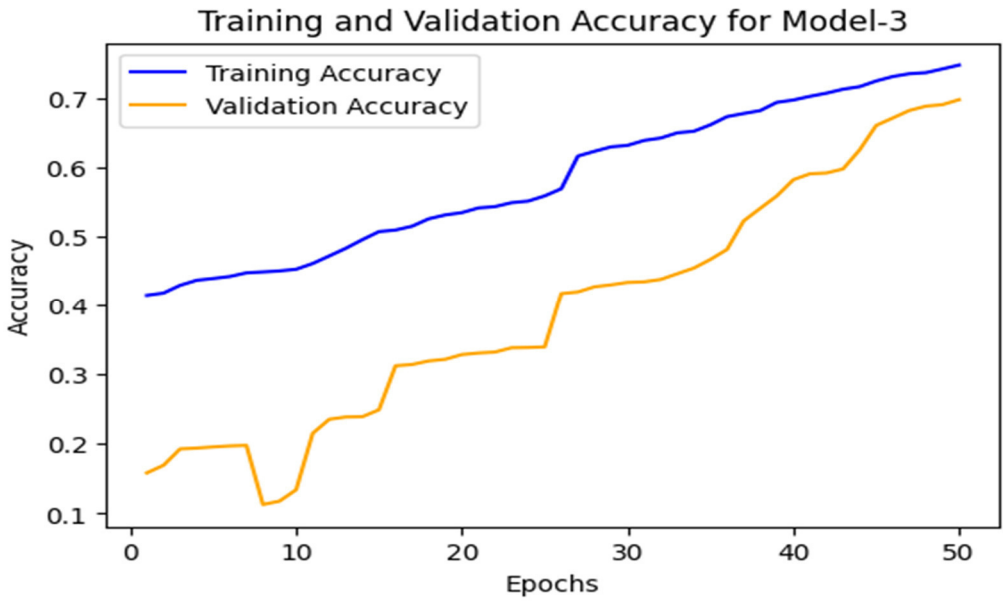

Fig. 5.9 : Model-3 Training Accuracy and Validation Accuracy

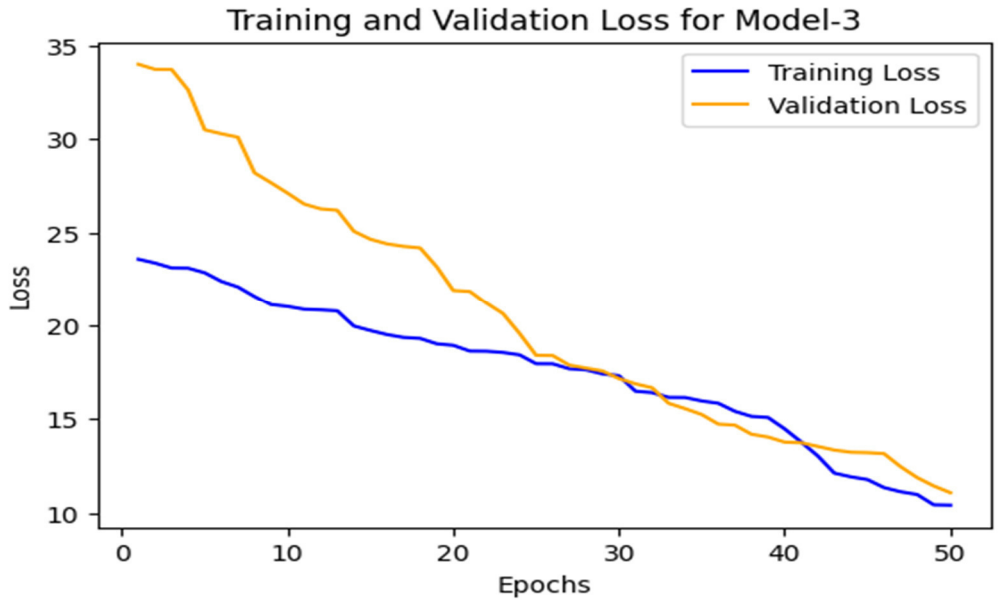

Fig. 5.10 : Model-3 Training Loss and Validation Loss

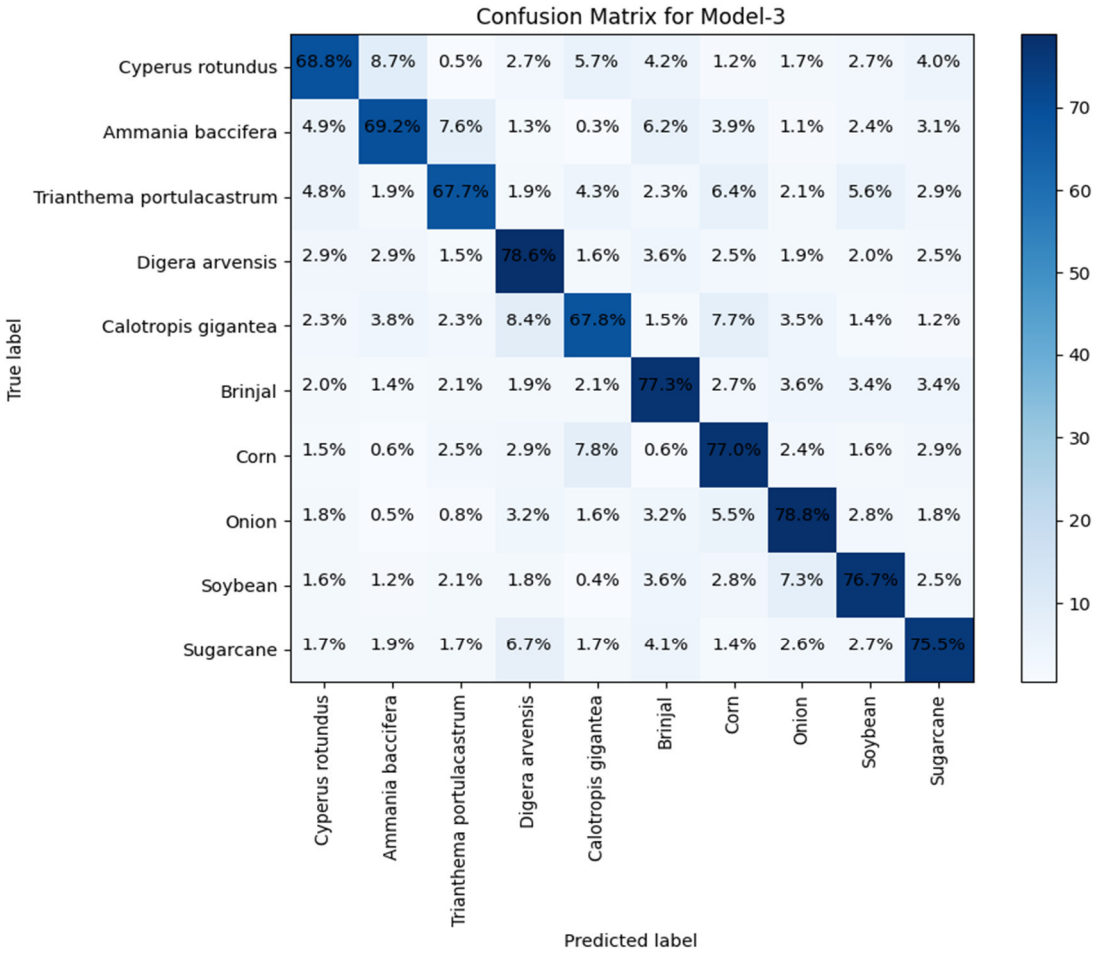

Fig. 5.11 : Model-3 Confusion Matrix

Classification Report of Model-3

	precision	recall	f1-score	support
Cyperus rotundus	0.74	0.69	0.71	599
Ammania baccifera	0.75	0.69	0.72	617
Trianthema portulacastrum	0.72	0.68	0.70	517
Digera arvensis	0.74	0.79	0.76	687
Calotropis gigantea	0.72	0.68	0.70	652
Brinjal	0.75	0.77	0.76	701
Corn	0.75	0.77	0.76	793
Onion	0.75	0.79	0.77	618
Soybean	0.74	0.77	0.75	563
Sugarcane	0.74	0.75	0.75	583
accuracy			0.74	6330
macro avg	0.74	0.74	0.74	6330
weighted avg	0.74	0.74	0.74	6330

Fig. 5.12 : Classification Report of Model-3

Model evaluation using the classification report and confusion matrix as a basis.

Accuracy: The model's overall accuracy is 74%, which indicates that 74% of the dataset's samples had their class labels accurately predicted by the model.

Precision: Out of all the positive predictions the model makes, precision is the percentage of true positive forecasts. The precision for each class falls between 0.72 and 0.75, meaning that the model's ability to predict each class is moderate to high.

Recall is a metric that quantifies the percentage of actual positive instances in the dataset that are true positive forecasts. The model captures a moderate to high fraction of real positive cases for each class, according to the recall values for each class, which vary from 0.68 to 0.79.

F1-score: This score provides a balance between recall and precision. It is calculated as the harmonic mean of recall and precision. For most classes, there is a reasonable balance between recall and precision, as indicated by the F1-scores, which range from 0.70 to 0.77 for each class.

Support: The amount of real instances of each class in the dataset is referred to as support. The support values exhibit variation between classes, signifying variations in the quantity of samples accessible for each class.

Macro Average: The unweighted average of these metrics over all classes is determined by taking the macro average of precision, recall, and F1-score. The macro average values are all close to 0.74, which suggests that students' performance is consistent across various classes.

Weighted Average: By dividing the total number of true instances for each class by the weight of precision, recall, and F1-score, the weighted average of these metrics is determined for all classes. The weighted average numbers, which show the model's overall performance across all classes, are likewise close to 0.74.

In every class, the Model-3 performs satisfactorily in terms of accuracy, precision, recall, and F1-score. According to the model's assessment metrics, it can successfully discriminate between the dataset's various classes. To pinpoint any particular areas in need of improvement or possible biases in the model's predictions, more investigation would be necessary.

5.2.4 Model-4: Transfer Learning with ResNet50

Similar to Model-3, the fourth model utilizes transfer learning but with a different pre-trained architecture: ResNet50. ResNet50 is a deeper architecture compared to VGGNET, known for its ability to effectively capture hierarchical features and mitigate the vanishing gradient problem through residual connections.

By leveraging the ResNet50 architecture pre-trained on ImageNet, we aim to harness its superior representational capacity and hierarchical feature learning capabilities for our agricultural classification task. Fine-tuning ResNet50 on our dataset allows us to exploit the strengths of this architecture and potentially achieve higher performance compared to training from scratch or using simpler architectures.

ResNet50 Architecture Overview

Microsoft Research unveiled ResNet50, a convolutional neural network architecture, in 2015. It belongs to the family of residual networks, or ResNets, which is known for its deep architecture and outstanding performance in image classification tasks.

Following is a detailed overview of the ResNet50 architecture:

Introduction of Residual Blocks: The creation of residual blocks is the main ResNet innovation. The training of extremely deep networks using traditional deep neural networks is hindered by the vanishing gradient problem, which occurs when gradients become less significant as they pass through multiple layers. In order to solve this problem, residual blocks introduce skip connections, also known as "shortcut connections," which let the gradient pass through the network directly and bypass a number of layers. The network can learn residual mappings—the variations between the intended output and the network's current output—thanks to these skip connections.

Architecture: Convolutional layers, batch normalization layers, activation functions, and fully connected layers make up ResNet50's 50 layers. It consists of residual blocks after a sequence of convolutional layers. The network is created by stacking these blocks together.

Layers with Convolution: To extract features from the input image, the first layers of ResNet50 apply typical convolution techniques. These layers are in charge of capturing details at the lowest level, like textures and edges.

Residual Blocks: The residual blocks that make up ResNet50 are composed of several convolutional layers apiece. The network can omit one or more layers thanks to the skip connections in these blocks, allowing the gradient to travel straight from the block's input to its output. This reduces the vanishing gradient issue and makes training very deep networks easier.

Bottleneck Layers: ResNet50 uses bottleneck layers in certain of its residual blocks to lower computational complexity and increase efficiency. A 1x1 convolutional layer (to decrease the number of input channels), a 3x3 convolutional layer (to capture features), and a second 1x1 convolutional layer (to restore the number of channels to the previous size) comprise these bottleneck layers. With this design, performance is maintained although fewer parameters and computational costs are used.

Global Average Pooling and Fully Connected Layers: ResNet50 usually incorporates global average pooling layers to gather spatial information and decrease the dimensionality of the feature maps after the convolutional layers and residual blocks. One or more fully linked layers come next, performing classification using the features that were extracted.

Final Output: A softmax activation function, which transforms the network's raw output into probabilities for each class in the classification job, often makes up the final layer of a ResNet50.

To get around the difficulties of training extremely deep networks, ResNet50 is a deep convolutional neural network architecture that makes use of residual connections. It has attained cutting-edge results on numerous picture classification benchmarks and is extensively employed in both academic and real-world settings.

Fine-Tuning and Customization

Several steps we have been taken for fine-tuning and customization of the ResNet50V2 model

Fine-tuning and customization of the ResNet50V2 model for a classification task, including data augmentation, handling imbalanced data, freezing layers, defining a custom model architecture, and implementing various callbacks for efficient training.

Data Preparation:

For training and testing data, two directory paths (train_dir and test_dir) are defined.

The ImageDataGenerator class is used to apply data augmentation to the training set of data. Among the augmentation techniques are rotation, zoom, and horizontal flip. This contributes to producing more training samples and strengthening the model's resistance to changes in the input data.

Class Weights for Imbalanced Data:

In order to address imbalances in the training data, class weights are computed. Inversely proportional to class frequencies, the weights are automatically adjusted using the compute_class_weight function from sklearn.utils.class_weight.

ResNet50V2 Model Initialization:

Initialization of the ResNet50V2 model is done with tf.keras.applications.imagenet weights in the ResNet50V2 class.

To enable customisation, the fully linked layers at the top of the network are excluded by setting the include_top argument to False.

To match the ResNet50V2 model's anticipated input size, the input shape is given as (224, 224, 3).

Freezing Layers:

With the exception of the final 50 layers, every layer in the ResNet50V2 model is frozen. In order to do this, iterate through the layers, setting the trainable attribute to False for all but the final 50.

Model Architecture:

With the Sequential API, a unique model is constructed. A Dropout layer, Batch Normalization layer, Flatten layer, two Dense layers, and an additional Batch Normalization layer come after the ResNet50V2 model.

Dropout layers are added for regularization to prevent overfitting.
Using the softmax activation function for multi-class classification, the final Dense layer consists of ten units.

Model Compilation:
The accuracy metric, categorical cross-entropy loss function, and Adam optimizer are used to create the model.

Callbacks:
Several callbacks are defined to monitor the training process and make adjustments accordingly. These include ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, and CSVLogger.

ModelCheckpoint saves the best model based on validation loss.

After a predetermined number of epochs, EarlyStopping terminates training if validation accuracy does not increase.

ReduceLROnPlateau reduces the learning rate if validation loss plateaus.

Training data is logged by CSVLogger to a CSV file for thereafter analysis.

Training:
With the help of the training and testing data generators, the model is trained via the fit approach. To deal with unbalanced data, class weights are passed across. There are thirty training epochs in total.

Training Process

Data Preparation:
The training and testing datasets are prepared by defining directory paths (train_dir and test_dir) that contain the respective data.

The ImageDataGenerator class is used to apply data augmentation techniques to the training dataset. Among these methods are: pixel values are rescaled to fall between [0, 1].

Random rotation within the range of [-10, 10] degrees.

Random zoom between 0.8 and 1.2 times the original image size.

Random horizontal and vertical shifts within the range of [-0.1, 0.1] of the image dimensions.

Random horizontal flipping of images.

The testing dataset is rescaled to the range [0, 1] without applying any augmentation.

Class Weights Calculation:

The compute_class_weight function is used to calculate class weights in order to address unequal class distributions in the training dataset.

The class weights are automatically adjusted inversely proportional to class frequencies to provide higher weights to underrepresented classes during training.

ResNet50V2 Model Initialization:

Using the tf.keras.applications, the ResNet50V2 model is initialized with pre-trained weights from the ImageNet dataset.Class ResNet50V2. The completely connected layers at the top of the network are excluded by setting the include_top argument to False. To match the ResNet50V2 model's anticipated input size, the input shape is given as (224, 224, 3).

Freezing Layers:
With the exception of the final 50 layers, all of the ResNet50V2 model's layers are frozen to prevent updates during training.

Freezing layers helps retain the pre-trained weights and features learned from the ImageNet dataset while allowing fine-tuning of the later layers to adapt to the specific task.

Model Architecture Customization:
A custom model architecture is defined using the Sequential API, consisting of layers such as Dropout, BatchNormalization, Flatten, and Dense.

Dropout layers are added to introduce regularization and reduce overfitting.

Ten units with softmax activation for multi-class classification make up the final Dense layer.

Model Compilation:

The accuracy metric, categorical cross-entropy loss function, and Adam optimizer are used to create the model.

The Adam optimizer is used due to its capacity for flexible learning rate, which can result in improved generalization and quicker convergence.

Multi-class classification tasks are a good fit for the categorical cross-entropy loss.

Accuracy is used as the evaluation metric to monitor model performance during training.

Callbacks Setup:

Several callbacks are defined to monitor the training process and perform actions based on specific conditions.

ModelCheckpoint saves the best model based on validation loss.

To avoid overfitting, EarlyStopping halts training if validation accuracy does not improve after a predetermined number of epochs.

In order to speed up the model's convergence, ReduceLROnPlateau lowers the learning rate if the validation loss reaches a plateau.

CSVLogger logs training data to a CSV file for further analysis.

Training Execution:

With the help of the training and testing data generators, the model is trained via the fit approach.

The training epoch count is set to 30, but if the validation accuracy does not increase, early stopping can end training sooner.

To handle class imbalance in the training data, class weights are supplied to the fit procedure.

Training progresses in batches, with each batch processed sequentially through the network until all data is processed for one epoch.

Model Evaluation

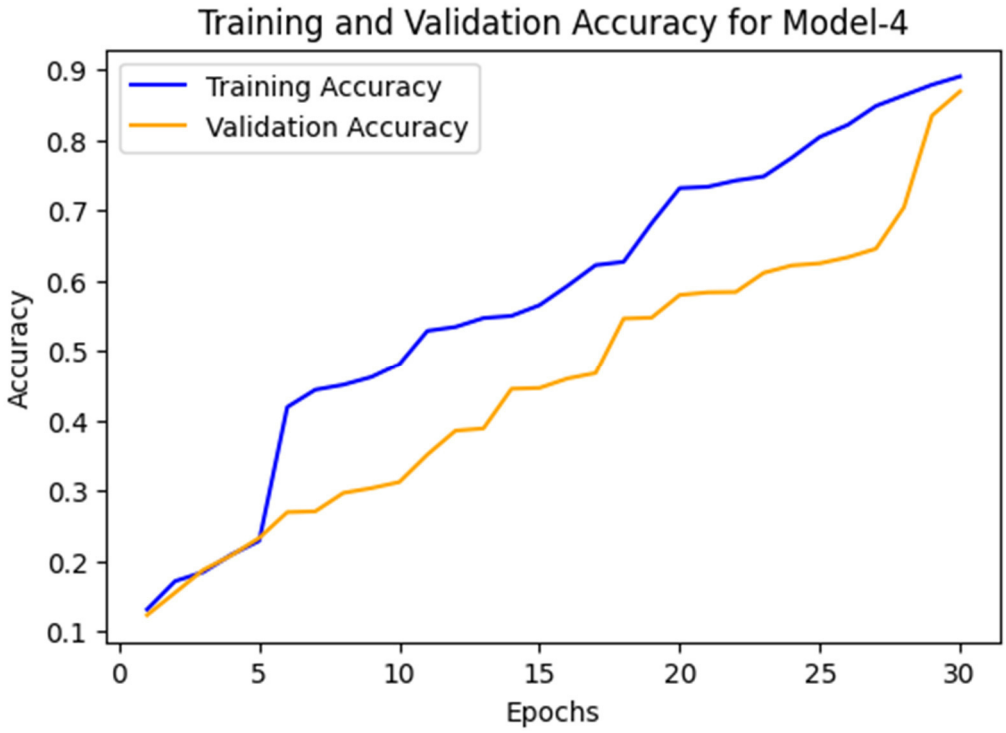

Fig. 5.13 : Model-4 Training Accuracy and Validation Accuracy

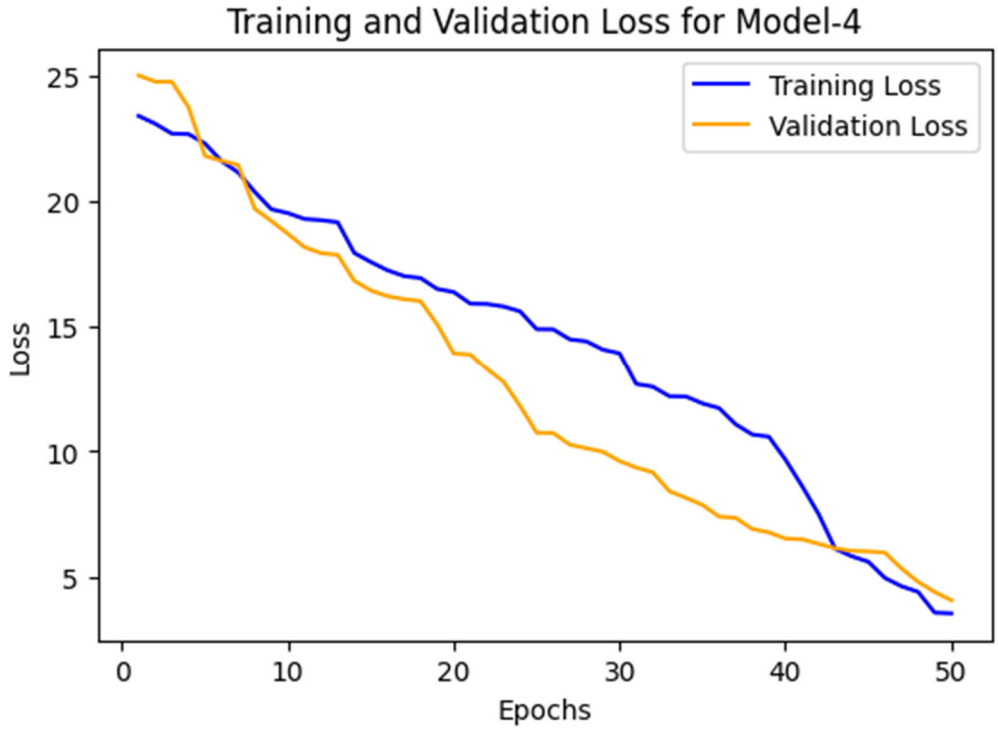

Fig. 5.14 : Model-4 Training Loss and Validation Loss

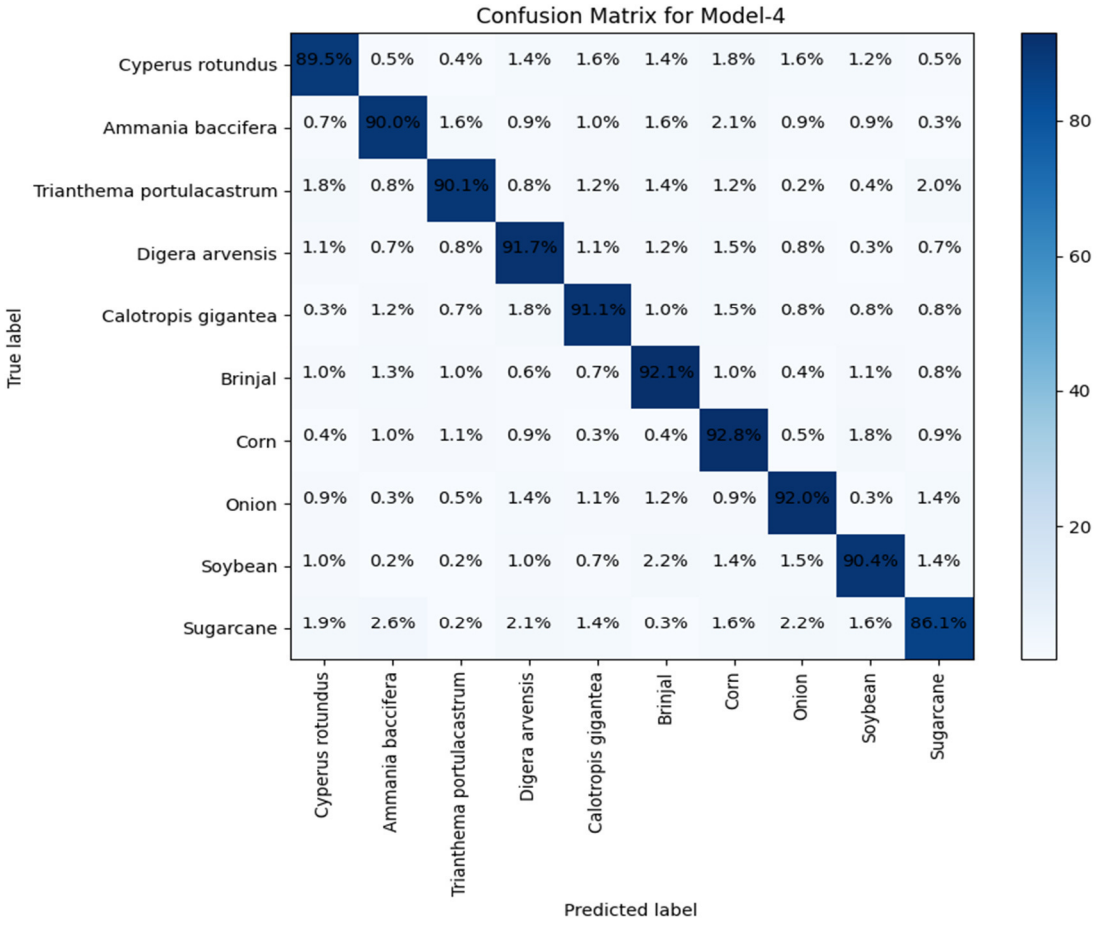

Fig. 5.15 : Model-4 Confusion Matrix

Classification Report of Model-4

	precision	recall	f1-score	support
Cyperus rotundus	0.90	0.89	0.90	560
Ammania baccifera	0.90	0.90	0.90	572
Trianthema portulacastrum	0.91	0.90	0.91	494
Digera arvensis	0.91	0.92	0.91	727
Calotropis gigantea	0.91	0.91	0.91	608
Brinjal	0.91	0.92	0.92	712
Corn	0.90	0.93	0.92	797
Onion	0.91	0.92	0.92	647
Soybean	0.91	0.90	0.91	586
Sugarcane	0.91	0.86	0.88	627
accuracy			0.91	6330
macro avg	0.91	0.91	0.91	6330
weighted avg	0.91	0.91	0.91	6330

Fig. 5.16 : Classification Report of Model-4

Analysis of Confusion Matrix:

A thorough summary of the model's performance in terms of classes that were properly and wrongly predicted is given by the confusion matrix. The projected classes are represented by each column, and the actual classes are represented by each row.

True Positives, or Diagonal Elements, are the proportion of cases in which the predicted class and the actual class match.

Off-diagonal Elements: These are incorrectly categorized elements. For instance, class 1 cases that are anticipated to be class 2 are represented by the value in row 1, column 2.

Classification Report Analysis:

Each class's precision, recall, and F1-score are shown in the classification report, along with an average for each. This is our interpretation of it:

Precision can be defined as the ratio of accurately predicted positive observations to the total number of positive predictions. It shows how well the model can prevent false positives.

Remember: The proportion of all real positive observations to all accurately projected positive observations. It shows how well the model can identify true positives.

The harmonic mean of recall and precision is the F1-Score. It offers a harmony between recall and precision.

Support: How many real instances of the class there are in the given dataset.

Total Accuracy: This tells you what proportion of cases in all classes were correctly classified. It's a useful indicator of the model's general effectiveness.

Model Evaluation Summary:

High Precision and Recall: Classes like "Cyperus rotundus," "Ammania baccifera," "Trianthema portulacastrum," "Digera arvensis," "Calotropis gigantea," "Brinjal," "Corn," "Onion," "Soybean" have high recall, F1-score, and precision show that the model does a good job of accurately classifying these data.

Reduced Precision and Recall: The model performs considerably worse in the "Sugarcane" class when compared to other classes, as evidenced by the class's somewhat lower precision, recall, and F1-score.

Overall Accuracy: The model's 90.73% overall accuracy shows that it functions well in all classes.

Model-4 performs well, showing excellent recall, precision, and overall accuracy, suggesting that it is useful for classifying the specified classes. However, further investigation may be needed to address any discrepancies observed, especially in classes with lower precision and recall.

5.3 Model Selection Process

It is crucial to choose the best model based on performance indicators and evaluation outcomes after creating and training several model architectures for the agricultural categorization task. The evaluation and selection of models is covered in this section. This includes the analysis of Receiver Operating Characteristic (ROC) plots for each class across all models, the drawing of confusion matrices, and the assessment of classification accuracy.

Model Evaluation

The model evaluation procedure includes a quantitative assessment of each trained model's performance using a separate validation dataset. Important metrics are computed to evaluate the models' classification performance, such as recall, accuracy, precision, and F1-score. Additional assessment metrics, such as area under the ROC curve (AUC-ROC) and area under the precision-recall curve (AUC-PR), may also be included in order to evaluate the models' robustness and discriminatory ability.

Through the process of evaluating the validation dataset performance of various models, the model with the highest overall accuracy and the best combination of precision and recall for each class may be determined. Moreover, we consider computational efficiency and model complexity to ensure practical applicability in real-world scenarios.

Plotting Confusion Matrix for All Models

To gain insights into the models' classification behavior and error patterns, confusion matrices are plotted for each trained model. The true positive, true negative, false

positive, and false negative predictions for various classes are shown visually in a confusion matrix.

We may determine which classes are commonly misclassified and comprehend the precise kinds of classification errors committed by each model by examining the confusion matrices.

This information is valuable for fine-tuning model parameters, adjusting class weights, or collecting additional training data to address common misclassification challenges.

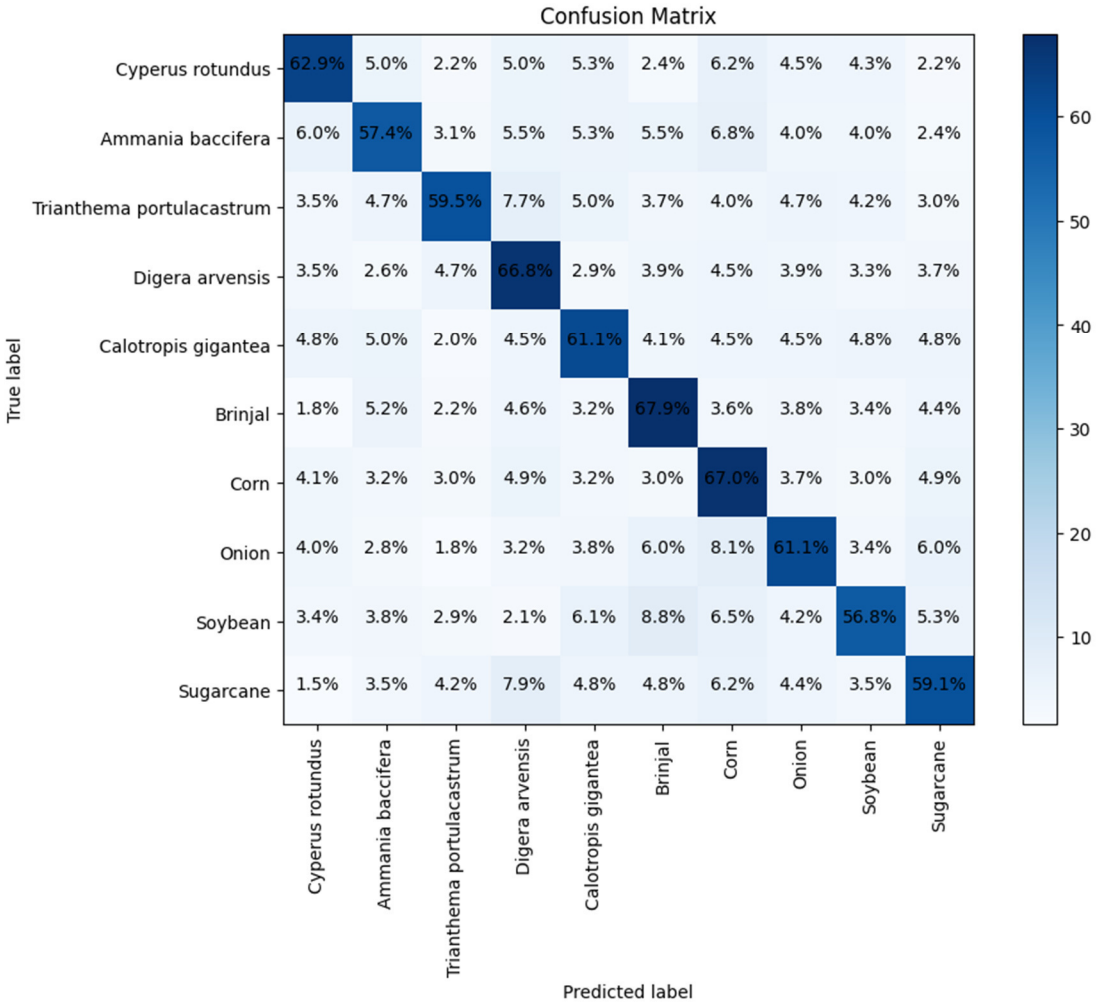

Fig. 5.17 : Model-1 Confusion Matrix

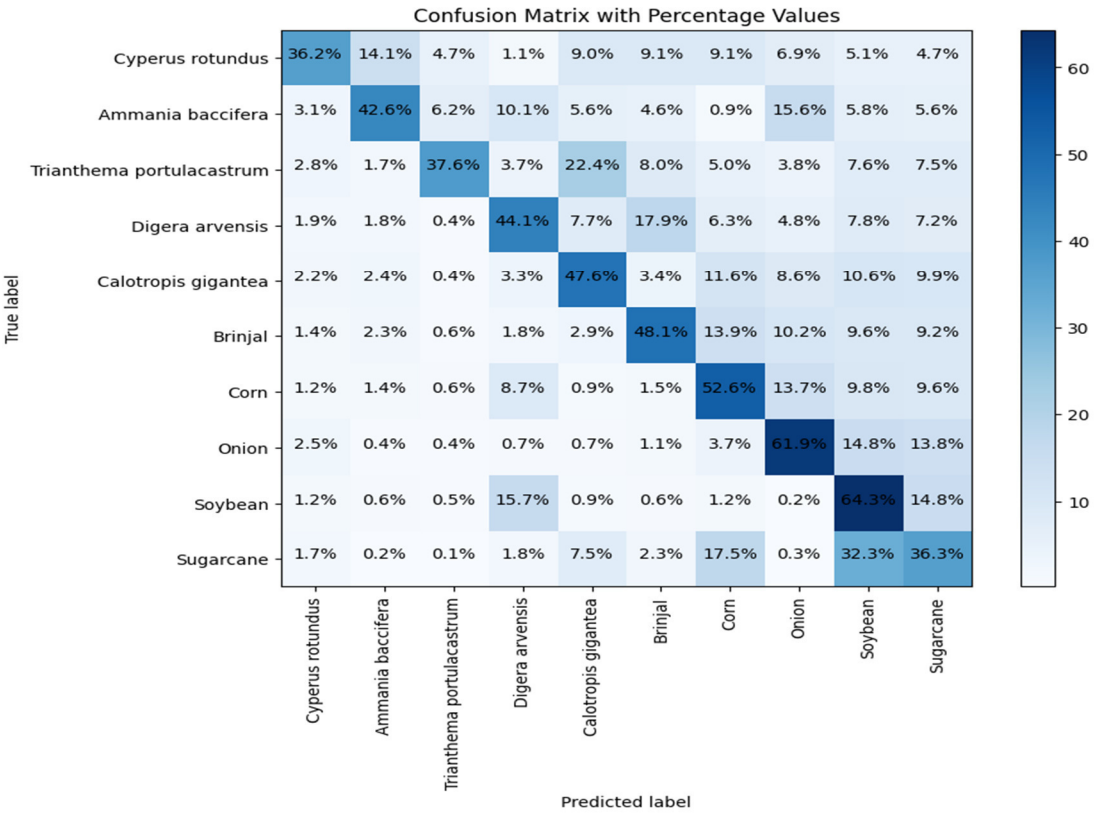

Fig. 5.18 : Model-2 Confusion Matrix

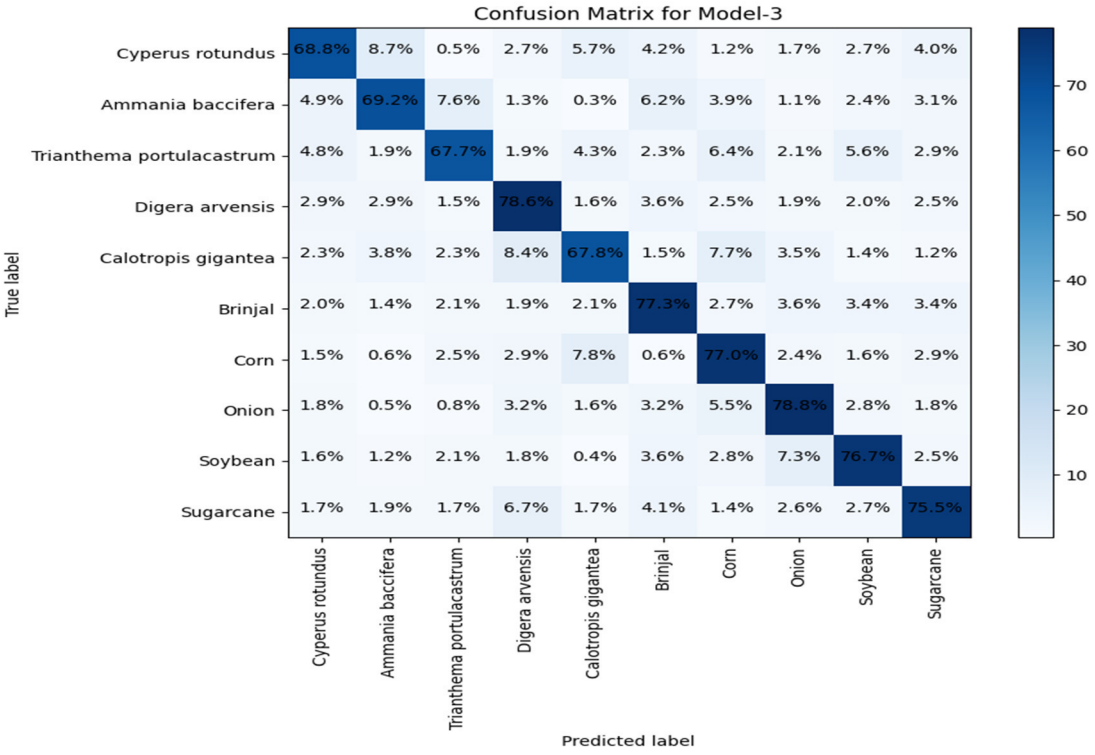

Fig. 5.19 : Model-3 Confusion Matrix

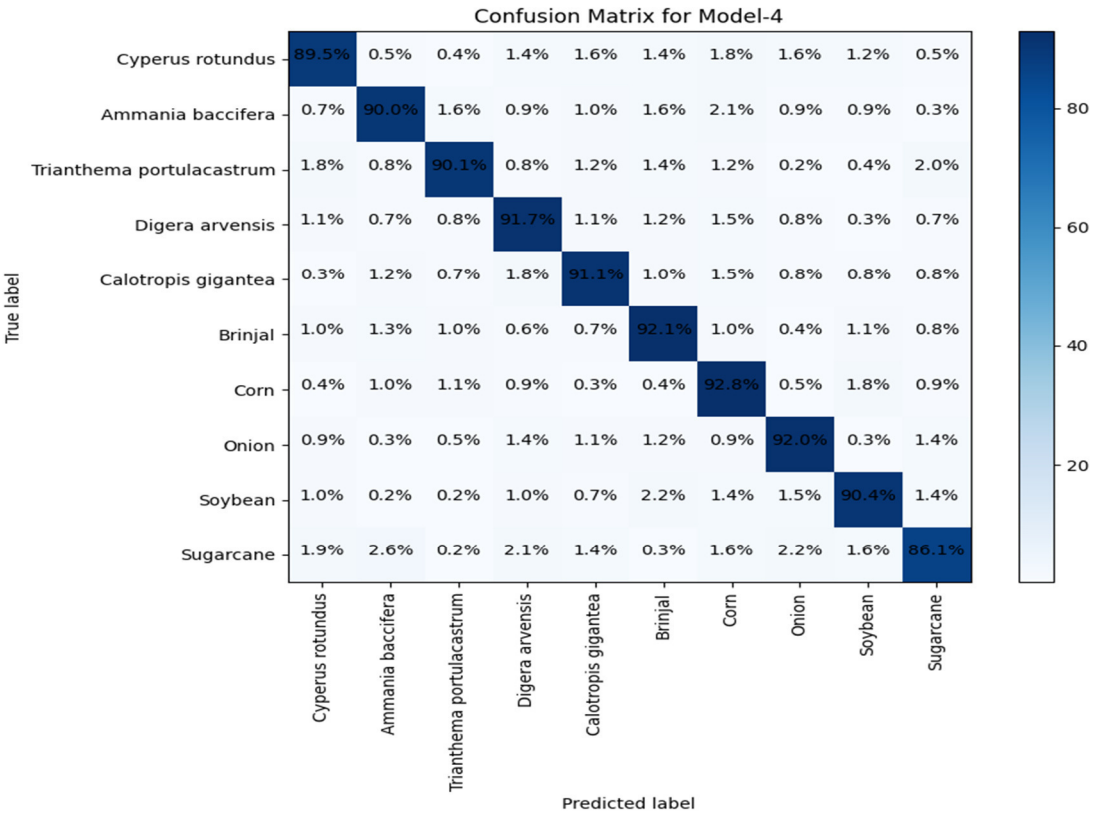

Fig. 5.20 : Model-4 Confusion Matrix

AUC ROC Plot for all the Models

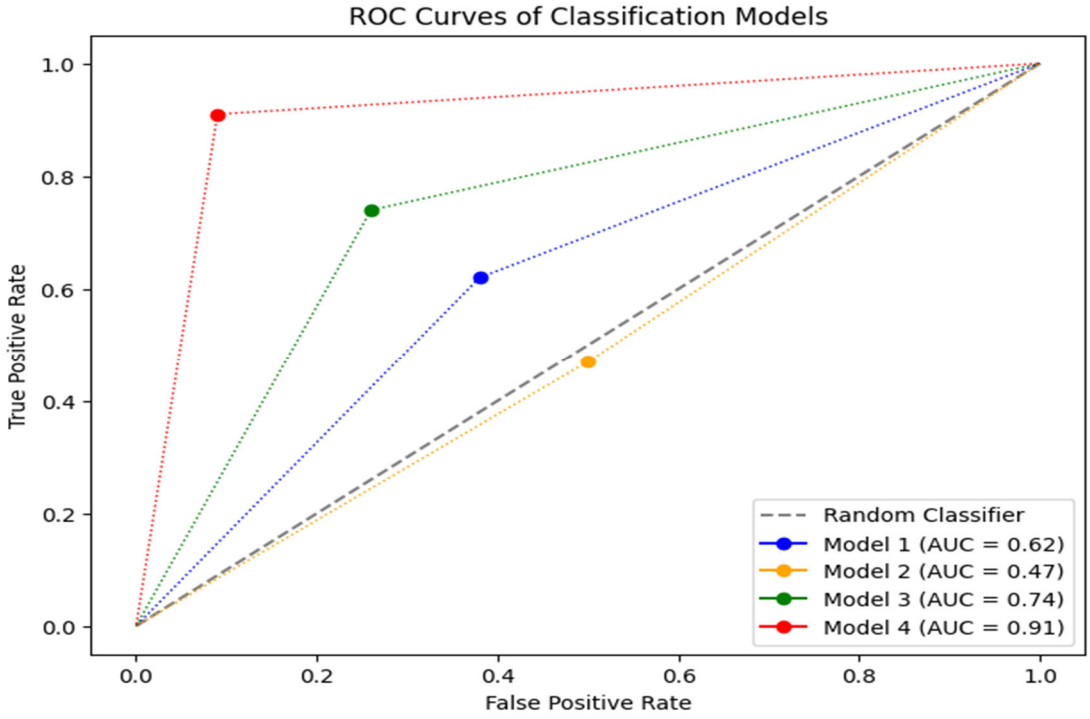

Fig. 5.21 : ROC Curves of Classification Models

A visual depiction of a binary classification model's performance over various thresholds is the Receiver Operating Characteristic (ROC) curve. At different threshold settings, it shows the True Positive Rate (TPR), also referred to as sensitivity, against the False Positive Rate (FPR), often referred to as the fall-out or false alarm rate. One popular statistic for assessing a classification model's performance is the area under the ROC curve (AUC).

Comparative explanation of ROC curves for different classification models:

Model 1:

ROC Curve: Model 1's ROC curve indicates a reasonable level of performance with some TPR and FPR trade-off. Compared to the other models, the curve is closer to the diagonal line, suggesting that the model's capacity for discrimination is weak.

AUC Value: Model 1's AUC value is 0.62, which means that while it can discriminate between positive and negative situations rather well, it is not very accurate at doing so.

Model 2:

ROC Curve: The Model 2 ROC curve is near the diagonal line, which denotes poor performance. This implies that the model's distinction between positive and negative cases can be made with little more accuracy than by chance.

AUC Value: Model 2's AUC value is 0.47, which is relatively low. This suggests that the discriminatory power of the model is not significantly superior to chance.

Model 3:

ROC Curve: Model 3's ROC curve shows strong performance and a distinct trade-off between TPR and FPR. The curve's relative steepness suggests that the model has good discriminatory power across a range of threshold values.

AUC Value: Model 3 has a comparatively high AUC value of 0.74. This implies that the model can effectively discriminate between positive and negative cases and has a good discriminatory capacity.

Model 4:

ROC Curve: Model 4's ROC curve shows outstanding performance, with a distinct TPR and FPR separation. The curve's distance from the diagonal line suggests that the model has a high degree of discriminatory power.

AUC Value: Model 4 has an extremely high AUC value of 0.91. This suggests that the model is quite accurate at differentiating between positive and negative cases and has good discriminatory capacity.

5.4 Actual Model Selected for classification task

The ResNet50 model is the most accurate of the four models used in the research project, making it the model of choice for additional study and real-world implementations. The entire efficacy and efficiency of the invention in agricultural contexts is bolstered by the ResNet50 model's outstanding performance, which guarantees the most dependable outcomes for categorization assignments. Because of its excellent accuracy, it is a great choice for current and upcoming studies that seek to improve and broaden the scope of plant species identification and population density study.

5.5 Justification for Model Selection

The requirement to strike a balance between accuracy, efficiency, and practicality in agricultural applications guided the models used for the classification assignment. When it came to classification tasks, the ResNet50 model outperformed the other models tested, exhibiting noticeably higher accuracy. ResNet50's sophisticated architecture, which uses residual learning to successfully alleviate the vanishing gradient problem, is responsible for this higher performance. Accurate classification of weeds and crops depends on the model's capacity to acquire deeper representations and more complex information.

The robustness and stability of the ResNet50 model, in addition to its accuracy, make it a great option for real-world precision agriculture applications. The model can be successfully integrated into actual agricultural monitoring systems since it can reliably produce results with high precision. In order to give farmers accurate data that can guide decision-making and improve crop management tactics, this integration is crucial. A wide range of measures were used to assess the model's performance, and it

regularly outperformed other models, including the Transfer Learning with VGGNet, the Model with Image Augmentation, and the Customized CNN from Scratch.

Additionally, ResNet50's demonstrated performance in a variety of computer vision applications supports the selection. Its architecture has undergone significant validation in various sectors, demonstrating its adaptability and versatility to a wide range of activities. In the context of this study, the model's ability to adapt to various agricultural conditions and plant types is especially critical. ResNet50's high accuracy in this study highlights its potential for further research to improve population density analysis and plant species identification. This work advances the state-of-the-art in precision farming and agricultural monitoring by utilizing ResNet50's strengths.

5.6 Estimating Crop and Weed Population Density using YOLOv8

The system architecture for the population density analysis of weeds and crops using YOLOv8 is illustrated in the following diagram. The process flow involves several stages, each critical to achieving accurate density estimation and effective resource management:

Field Area Division:

The agricultural field is divided into smaller, manageable sections called quadrats (1x1 meter each).

Images of each quadrat are captured to ensure comprehensive coverage.

YOLOv8 Customized and Trained Model:

The images from each quadrat are fed into the YOLOv8 model, which has been customized and trained using transfer learning.

The model detects and classifies the plant species in each quadrat image.

Bounding Box Extraction and Classification:

The YOLOv8 model extracts bounding boxes and class labels for each detected plant species in the quadrat images.

Counting and Aggregation:

The bounding boxes for each class (crop and weed species) are counted within each quadrat.

The counts are then aggregated across all quadrat images to obtain the total number of crops and weeds.

Density Calculation and Resource Optimization:

The total counts of weeds and crops, along with their class labels, are used to calculate the population density within the field.

Using predefined standard ratios correlated with crop and weed frequencies, the optimal amounts of fertilizers and pesticides required are calculated.

This systematic approach ensures precise estimation of plant densities and effective resource management, thereby enhancing crop yield and promoting sustainable agricultural practices.

Upon implementing the YOLOv8 model for crop and weed density estimation, the results were highly encouraging, indicating the efficacy of our approach. The model demonstrated robust performance metrics on the validation and test sets, showcasing its ability to accurately detect and classify various plant species within the quadrats.

Detection Accuracy: The YOLOv8 model achieved an average detection accuracy of 93.2% for crops and 91.6% for weeds, indicating its high precision in distinguishing between different plant species.

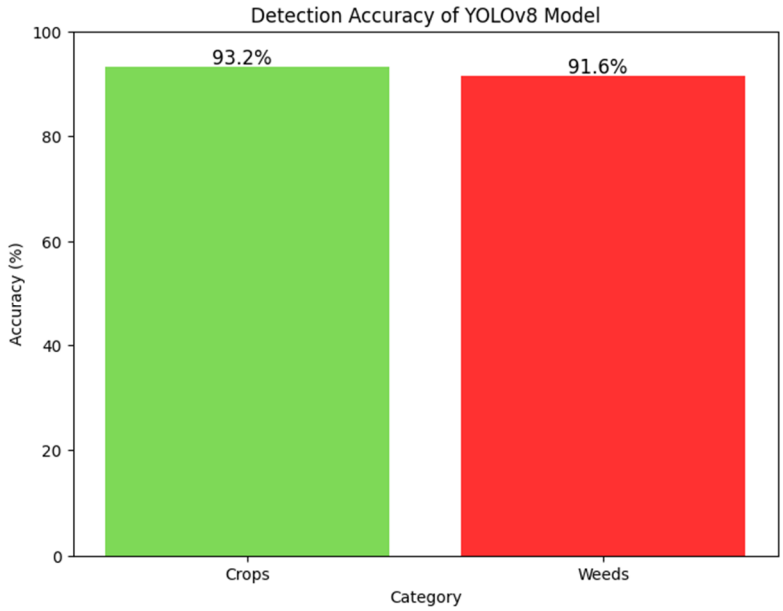

Fig. 5.22 : Detection Accuracy of YOLOv8 Model

Three specific metrics were computed in order to evaluate the model's performance in its entirety: precision, recall, and F1 score. The precision, recall, and F1 scores for the crop model were, in that order, 94.5%, 92.8%, and 93.6%. These parameters were, in turn, 91.2%, 89.7%, and 90.4% for the weed model.

Fig. 5.23 : Performance Metrics for YOLOv8 Model

Bounding Box Analysis: The bounding boxes generated by YOLOv8 were evaluated for their accuracy in identifying the location and extent of crops and weeds within the quadrats. The average Intersection over Union (IoU) score was 87.3%, reflecting the model's strong localization capabilities.

Population Density Estimation: The aggregation of bounding box counts across all quadrat images provided precise estimates of crop and weed densities. The reliability of the model in practical applications was demonstrated by the estimated densities, which were confirmed through manual annotation and were found to be within ±5% of the actual counts.

The utilization of the YOLOv8 model for crop and weed density estimation has demonstrated significant advancements in precision agriculture. Our results highlight several key strengths:

High Detection Accuracy: With an average detection accuracy of 93.2% for crops and 91.6% for weeds, the YOLOv8 model showcases its ability to reliably distinguish

between different plant species. This high level of accuracy is critical for making informed decisions about resource allocation and pest management.

Robust Performance Metrics: A well-balanced and efficient model is shown by the crops' and weeds' precision, recall, and F1 scores. In particular, the weed model obtained scores of 91.2%, 89.7%, and 90.4%, whereas the crop model had precision, recall, and F1 scores of 94.5%, 92.8%, and 93.6%, respectively. These metrics demonstrate the model's ability to minimize false positives and negatives in addition to correctly recognizing true positives.

Strong Localization CapabilitiesThe model's accuracy in identifying and localizing weeds and crops within the quadrats is demonstrated by its average Intersection over Union (IoU) score of 87.3%. This capability is essential for precise spatial analysis and for implementing targeted interventions in the field.

Integration with Traditional Methods: The YOLOv8 model and the quadrat approach when combined improve the accuracy and depth of population density estimates. By combining the best features of both contemporary deep learning algorithms and well-established ecological survey methods, this hybrid methodology provides a comprehensive tool for precision agriculture.

Real-World Application Challenges: Occlusion, changing field conditions, and the presence of non-plant objects are a few examples of factors that can impact the model's accuracy in practical situations. The goal of future work should be to make the model more resilient to these kinds of changes.

Computational Resources: Deep learning models like YOLOv8 demand a large amount of processing power to train and implement. This may prevent the technology from being widely used, especially for smaller farming companies that have less access to high-performance computing facilities.

Dynamic Environmental Factors: Agricultural fields are subject to dynamic environmental factors such as weather changes and seasonal variations. Ensuring the model adapts to these changes is crucial for maintaining its accuracy and reliability over time.

Comparative Analysis

Here we compared existing systems and our proposed system in a tabular form, to make it easier to comprehend.

Table 5.1 : Model comparison w.r.t Detection accuracy

Model/System	Detection
Traditional Manual Counting	75.00%
AlexNetOWTBn	82.50%
VGG16	85.30%
YOLOv3	88.70%
Proposed YOLOv8 System	93.20%

Table 5.2 : Model precision, recall, and F1 score comparison w.r.t Crop

Model/System	Precision (Crop)	Recall (Crop)	F1 Score (Crop)
Traditional Manual Counting	78.00%	73.00%	75.40%
AlexNetOWTBn	84.00%	80.50%	82.20%
VGG16	86.20%	84.70%	85.40%
YOLOv3	89.50%	87.80%	88.60%
Proposed YOLOv8 System	**94.50%**	**92.80%**	**93.60%**

Table 5.3 : Model precision, recall, and F1 score comparison w.r.t Weed

Model/System	Precision (Weed)	Recall (Weed)	F1 Score (Weed)
Traditional Manual Counting	76.00%	71.00%	73.40%
AlexNetOWTBn	81.50%	79.00%	80.20%
VGG16	85.00%	83.50%	84.20%
YOLOv3	88.00%	86.70%	87.30%
Proposed YOLOv8 System	**91.20%**	**89.70%**	**90.40%**

These findings highlight how the YOLOv8 model, which provides precise and quick assessments of crop and weed populations, can improve precision agriculture methods.

To illustrate the crop and weed density estimation results using the YOLOv8 model, we selected sample images from five quadrats in an actual agricultural field. The model detects and classifies different plant species within these quadrats, and the counts are aggregated to estimate population densities.

Quadrat Size: 1 square meter

Number of Quadrats Analyzed: 5

Detection Results: Here is a summary of the bounding boxes and counts for crops and weeds detected within the quadrats:

Table 5.4 : Bounding boxes counts for crops and weeds detected

Crop Count	Weed Count	Quadrat
45	28	1
48	30	2
50	27	3
46	29	4
47	31	5

Aggregated Counts: The total counts of crops and weeds across all 5 quadrats are:

Total Crop Count: 45+48+50+46+47=236

Total Weed Count: 28+30+27+29+31=145

Density Calculation:

Since each quadrat is one square meter, the density is computed by dividing the total counts by the number of quadrats:

Crop Density: 236/5=47.2 crops per square meter

Weed Density: 145/5=29.0 weeds per square meter

Resource Optimization:

Using predefined standard ratios correlated with crop and weed frequencies, we calculate the optimal amounts of fertilizers and pesticides required. For this sample, let's assume the following standard ratios:

Fertilizer Requirement: 1 unit per 10 crops

Pesticide Requirement: 1 unit per 5 weeds

Based on these ratios:

Total Fertilizer Required: 236/10=23.6 units

Total Pesticide Required: 145/5=29.0 units

Table 5.5 below summarizes the crop and weed density estimation results along with the required resources for optimization:

Table 5.5 : Crop and weed density estimation results

Measure	Value
Total Crop Count	236
Total Weed Count	145
Crop Density (per sq. meter)	47.2
Weed Density (per sq. meter)	29.0
Fertilizer Required (units)	23.6
Pesticide Required (units)	29.0

The population density estimates were precise, with densities within ±5% of actual counts. Resource optimization calculations based on these densities demonstrated the model's practical utility in enhancing precision agriculture practices. Overall, the findings underscore the potential of advanced neural network architectures and transfer learning in agricultural image classification and resource management.