Plant Seedlings Classification Using Deep Learning

Mrs. G. Bharathi[1], Sk. Farheen[2], Sk. Ashrita parvin[3], U. Rajarajeswari[4], Y.Nikhila[5]

[1] Sr.Gr.Asst.Professor, Department of CSE, SRGEC, Gudlavalleru,
[2,3,4,5] Undergraduate Student, Department of CSE, SRGEC, Gudlavalleru.

Abstract. Agriculture is crucial to human survival and continues to be a major economic force worldwide, particularly in undeveloped and developing nations. There is a need to enhance plant production while lowering costs because of the growing global population and the difficulties caused by climate change. Weed control plays a crucial role in this scenario. Identifying weeds is one of the most crucial tasks after a few days of plant germination because it allows farmers to implement early-stage weed control to decrease negative effects on crop growth. Classification of seedlings is important in the agriculture and botany fields. This study uses a dataset of about 4750 pictures of 12 species (9 weeds and 3 crops) in various developmental stages to offer an apprssssoach for categorising plant seedlings. Several classification techniques are proposed in this project to forecast the plant sort outcome from the test data, which include CNN (convolutional neural networks), ResNet50 (residual networks), and Xception models. From the comparison of the models, Xception with a 300x300 pixel size generated the best results with an average F1 score of 97% and an accuracy score of 97%, which demonstrates that this deep learning model can accurately categorise 12 different plant and weed seedlings in their early growth phases and that farmers may find this tool beneficial in weed identification.

Keywords: Resnet50, Convolutional neural network, Xception, and Plant seedling classification.

1 Introduction

Agriculture is a vital industry globally, employing many individuals and providing the world's food supply. As the world's population increases, the demand for food rises, making it difficult for farmers to boost productivity. One of the biggest challenges they face is weed control. Weeds can significantly harm agricultural crop production by quickly capturing scarce natural resources such as light, water, soil nutrients, and space. They have traits like deep roots, drought and frost resilience, and high nutrient usage efficiency, which enable them to reproduce more quickly than cultivated plants. Additionally, weeds can produce compounds known as allelopathics that can promote the growth of pests and crop diseases. Therefore, it is crucial to detect and remove weeds from fields as soon as possible to avoid crop loss. While herbicides can control weeds, they can also be costly, environmentally damaging, and require a lot of labor. Moreover, when herbicides are not available, farmers face significant issues, including difficulty in trimming grass if weeds are left unchecked.

Weeds and plants can be challenging to differentiate, especially in their early stages. A possible solution to these challenges is to automatically detect weeds early on. Deep learning techniques can be utilized to accurately, cost-effectively, and safely classify and identify weeds and crops.

The plant seedling classification was performed in this study using various deep learning techniques, including a self-built CNN model, ResNet-50 as a feature extractor, a finely tuned pre-trained ResNet-50 model, and Xception as a feature extractor model. Each model performance was evaluated for the accurate classification of plant and weed seedlings. This project aims to determine the optimal architecture capable of distinguishing between weeds and plants accurately, leading to the development of an automated system that can be applied in the field.

2 Literature Review

The authors of [1] developed a neural network that is trained from scratch. The resized 224 x 224 images are fed to the network. The network contained seven layers, out of which five are convolutional layers and two are fully connected layers. The performance of classifying seedling images from 12 different species is investigated. The model achieved an accuracy rate of 90.15%.

To identify various plant species depicted in colored photographs, the authors of [2] developed a deep convolutional neural network from scratch. The network was trained on 128x128 RGB images and consisted of a convolutional layer, a max pooling layer, and two residual blocks, each containing two convolutional layers and one max pooling layer. ReLU and batch normalization were linked to each convolutional layer, respectively. Finally, the network contained two categorization layers. The network was tested and trained on 10,413 images depicting 22 different types of crops and weeds, achieving an accuracy of 86.2% in classification.

The authors of [3] used a sizable, 17,509 image mosaic dataset that was made available to the public. Images were gathered from various Northern Australian regions and divided into eight types. Higher accuracy for ResNet-50 was achieved after implementing CNN architectures.

In order to distinguish between crops and weeds, the authors in [4] created methods employing binary image conversion and SVM and achieved an accuracy of about 50%.

Resnet, VGG16, and Inception V3 deep learning architectures were employed by the authors of [5] to detect banana diseases. Resnet-152 has achieved high accuracy for the given dataset.

Convolutional neural networks[6] (CNNs) are a type of deep learning model designed specifically for image classification. CNNs are composed of several layers, including a feature extraction network and a classifier network. The feature extraction network is made up of multiple pairs of convolutional layers and pooling layers, which work together to extract important features from the input images. These features are then passed on to the classifier network for further processing and classification[7].

Although CNNs are currently a popular technique in deep learning for image classification, they are not entirely new. In fact, as far back as the 1990s, CNNs were already being used for handwritten digit classification[8].

Krizhevsky et al.'s implementation of a wider and deeper CNN, known as AlexNet[9], marked a turning point in the success of CNNs for image classification in 2012. Since then, there have been significant advancements in the field of image classification achieved through the use of even deeper CNNs[10]-[12].

Although transfer learning has demonstrated impressive accuracy in classifying plant species, most of the existing studies have focused exclusively on fully grown plants, with only a few examining the classification of plant and weed seedlings. One such study was conducted by M. Dyrmann, H. Karstoft, and H. S. Midtiby,[13] which investigated the use of transfer learning for classifying plant and weed seedlings.

The Residual Network (ResNet)[14], which won the ILSVRC 2015 competition, was developed by a team at Microsoft led by K. He[12].

In CNNs, the Rectified Linear Unit (ReLU) [15]non-linearity is typically utilized to process the output of every convolutional and fully-connected layer.

3 Methodology

This section outlines our method from dataset collection to plant and weed seedling image classification using deep learning. The flow of our proposed approach is depicted in Fig. 1.

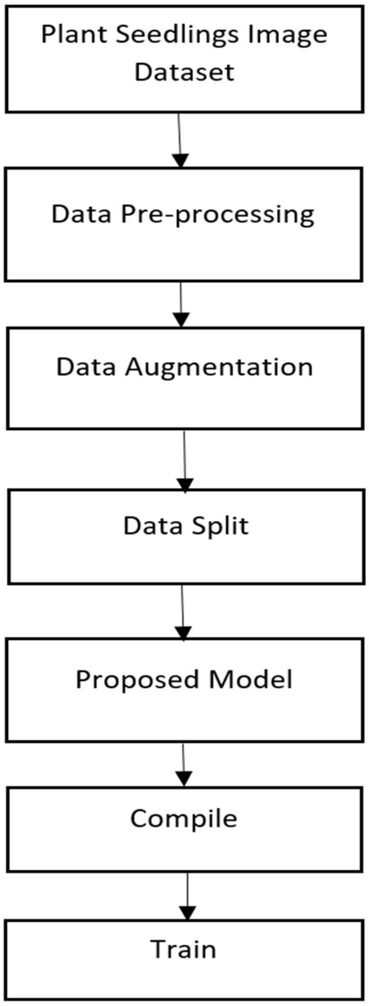

Fig. 1. Flow Chart of proposed method

3.1 Dataset

To evaluate the effectiveness of the architectures investigated in this study, a publicly available dataset from Kaggle is used, which has approximately 5544 pictures, of which 794 images represent the test set and 4750 images represent the training set. The images depict distinctive plants from 12 different plant species. Table 1. lists the 12 species of plant and weed seedlings.

Table 1. Number of Classes and Instances Used in Experiment

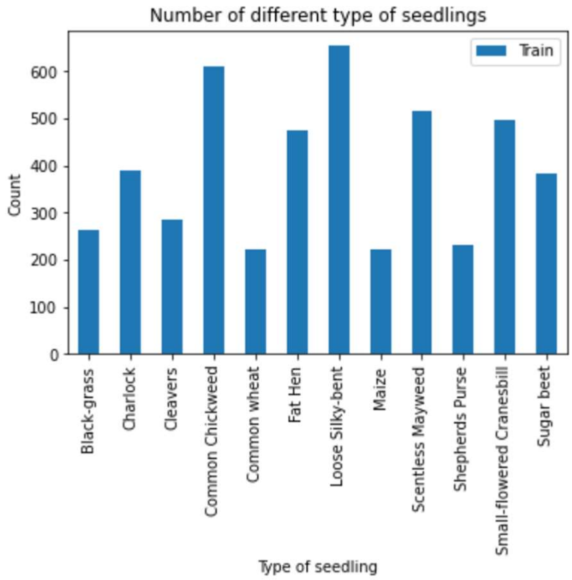

Fig. 2. Bar Graph indicating the proportion of each species in the train dataset.

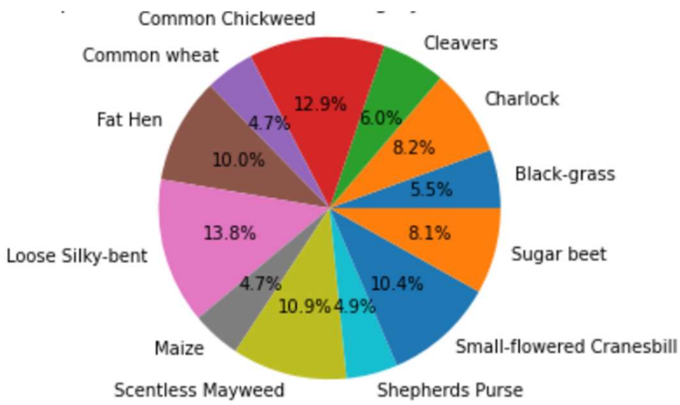

Fig. 3. Pie chart indicating Proportion of each species in the train dataset.

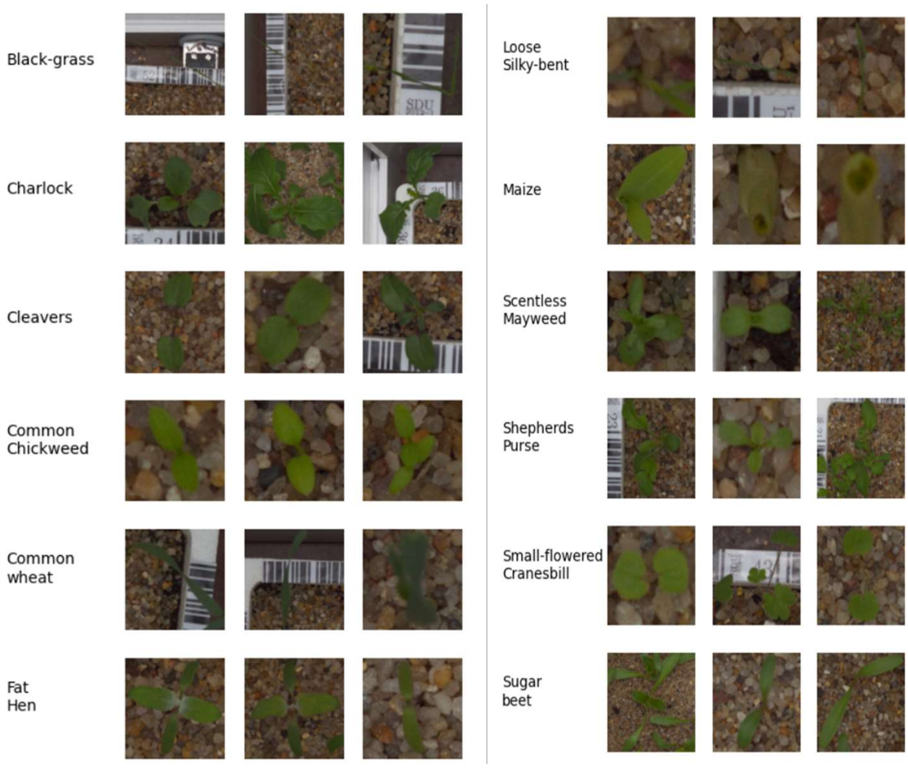

Fig. 4. Some of the plant and weed seedling images.

3.2 Image Pre-processing

The pictures are resized to 224 x 224 pixels and 300 x 300 pixels. Pixel values are normalized as per the model, which gives a better training result.

3.3 Data Augmentation:

A data augmentation technique is performed to reduce over-fitting during training. This procedure is very helpful when dealing with an unbalanced dataset. The augmentation methods applied are: rotation by 180 degrees; zooming in a range of 0–30%; width-shift and height-shift in a range of 0–30%; horizontal and vertical flips.

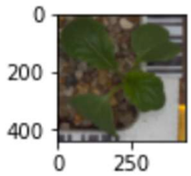

Fig. 5. Actual image of Charlock

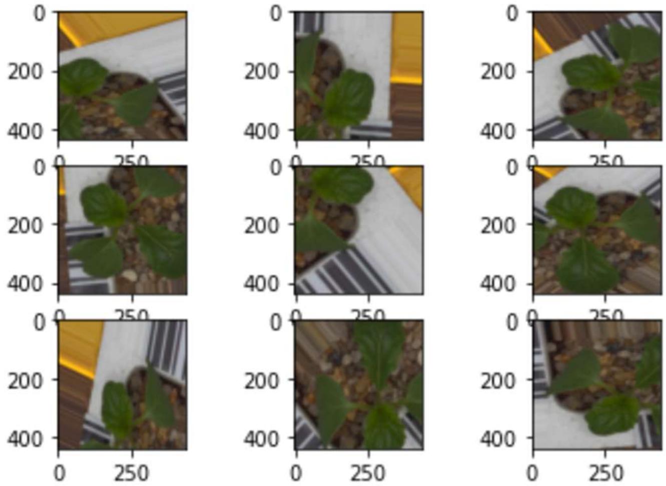

Fig. 6. Some of the images generated after performing augmentation on figure 5

3.4 Data Split:

The dataset includes a pre-existing training set and test set. The training set is partitioned into a 9:1 ratio, where 90% of the images are utilized for training and the remaining 10% for validation. Following the split, the training set, validation set, and test set consist of 4279, 471, and 794 images, respectively, that belongs to 12 distinct classes.

3.5 Model Building

In this project, several models (classifiers) are employed: a self-built CNN model, a ResNet-50 model, and an Xception model.

Convolutional Neural Network. Convolutional Neural Network (CNN) works better with images since it can discover features by itself. A CNN employs 2D convolutional layers and integrates input data with learned characteristics, making it an excellent architecture for processing 2D data, such as pictures.

A convolutional neural network usually comprises of an input layer that accepts input data, one or more hidden layers that perform convolutions, pooling, and nonlinear activations, and an output layer that provides the final prediction based on the processed input data.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 dropout (Dropout)           (None, 111, 111, 32)      0

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 54, 54, 64)        0

 conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 26, 26, 128)       0

 conv2d_3 (Conv2D)           (None, 24, 24, 256)       295168

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 256)      0
 2D)

 dropout_3 (Dropout)         (None, 12, 12, 256)       0

 flatten (Flatten)           (None, 36864)             0

 dense (Dense)               (None, 256)               9437440

 dense_1 (Dense)             (None, 256)               65792

 dense_2 (Dense)             (None, 12)                3084

=================================================================
Total params: 9,894,732
Trainable params: 9,894,732
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. Summary of layers in CNN model

CONV layer: The CONV layer computes the output of neurons that are connected to specific local regions within the input volume. Each neuron calculates a dot product between its weights and the corresponding small region of the input volume to which it is connected.

POOL layer: The pool layer produces a smaller volume than the preceding layer by down sampling along the spatial dimensions (width and height). These are employed to lessen computational expense and reduce overfitting to some extent.

Dense layer: A dense layer, also referred to as a fully connected layer, connects every neuron in the previous layer to each neuron in the dense layer. Each node within the dense layer generates a score value that represents a class score.

Dropout layer: Dropout layer is used as a technique for regularizing the training set's overfitting. It makes CNN less complex after each iteration by "dropping" neurons at random (by setting weights to zero), which also makes it difficult for the model to over-fit.

Transfer Learning. The phrase "transfer learning" refers to applying a model that has already been trained to a different problem. Transfer learning is the process through which a computer enhances its prediction of a new task using the knowledge it has learned from a previous task.

ResNet50. The residual network is known as ResNet. ResNet has numerous variations that use the same idea but have various numbers of layers. Resnet50 is the name given to the version that supports 50 neural network layers. It has forty-eight convolution layers, one average pool layer, and one MaxPool layer. The floating-point operation count is 3.8 x 109. It's a common ResNet model.

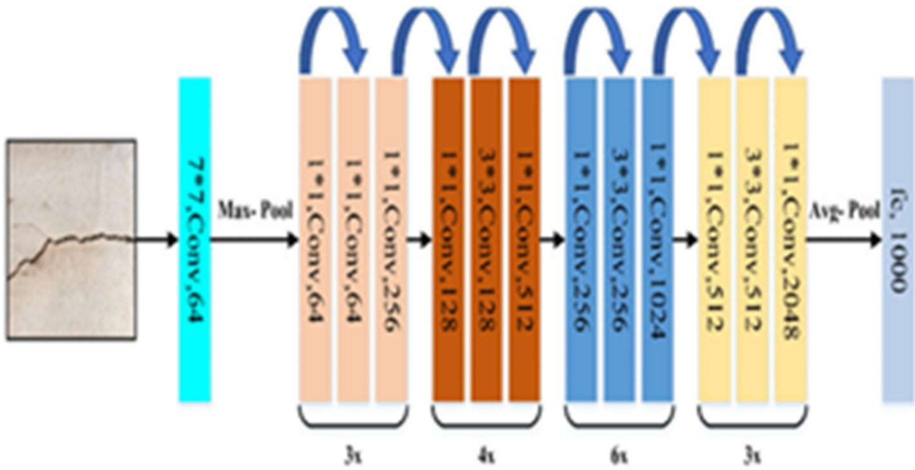

Fig. 8. ResNet-50 Architecture

ResNet-50 as feature extractor. In this technique, all the convolutional blocks are frozen, and just the fully connected layers are trained on the fresh dataset. In practice, this equates to employing a specific layer or layers of the pre-trained ResNet-50 as a fixed feature extractor.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 128)               65664

 dense_3 (Dense)             (None, 12)                1548

=================================================================
Total params: 26,277,900
Trainable params: 2,690,188
Non-trainable params: 23,587,712
_________________________________________________________________
```

Fig. 9. Summary of layers in Resnet50 as feature extractor model.

The model was trained using ResNet50, as illustrated in Fig. 9. A pre-trained network, which comes with weights that are trained on a large dataset, serves as the base model. The fully connected layers of the pre-trained model were replaced to adapt to the specific categorization problem. The fully connected classifier consisted of four layers, each comprising of 1024, 512, and 128 nodes with ReLU activation function, and a layer with 12 nodes that uses Softmax activation function.

Fine-tuning the ResNet50. Other typical transfer learning strategy entails not only retraining the classifier on top of the network with the new dataset, but also fine-tuning the network by simply training the higher-level segment of the convolutional layers and continuing backpropagation.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 128)               65664

 dense_3 (Dense)             (None, 12)                1548

=================================================================
Total params: 26,277,900
Trainable params: 8,210,572
Non-trainable params: 18,067,328
_________________________________________________________________
```

Fig. 10. Summary of layers in ResNet-50 finely tuned model.

In this work, lower-level layers of the network are frozen since they contain more generic features of the dataset. Only the top layers of the network are trained due to their ability to perform the extraction of more specific features. As shown in Fig. 10., only the last 4 layers are trainable, which are the dense layers having 1024, 512, and 128 units with activation function ReLU and a layer having 12 units with activation function ReLU.

Xception. The Xception, short for "extreme inception," is a concept that takes the principles of Inception to an extreme. In Inception, the initial input was compressed using 1x1 convolutions, and the depth space was then created from each of the input spaces using various types of filters. This step is simply reversed by Xception. Instead, it first filters each depth map individually before using 1X1 convolution to finally condense the input space across the depth. A depthwise separable convolution, an operation that was employed in neural network creation as early as 2014, is almost the same as this technique.

There is yet another difference between Inception and Xception. whether or not there is a non-linearity after the original procedure. While Xception doesn't introduce any non-linearity, the Inception model has a ReLU non-linearity that follows both procedures.

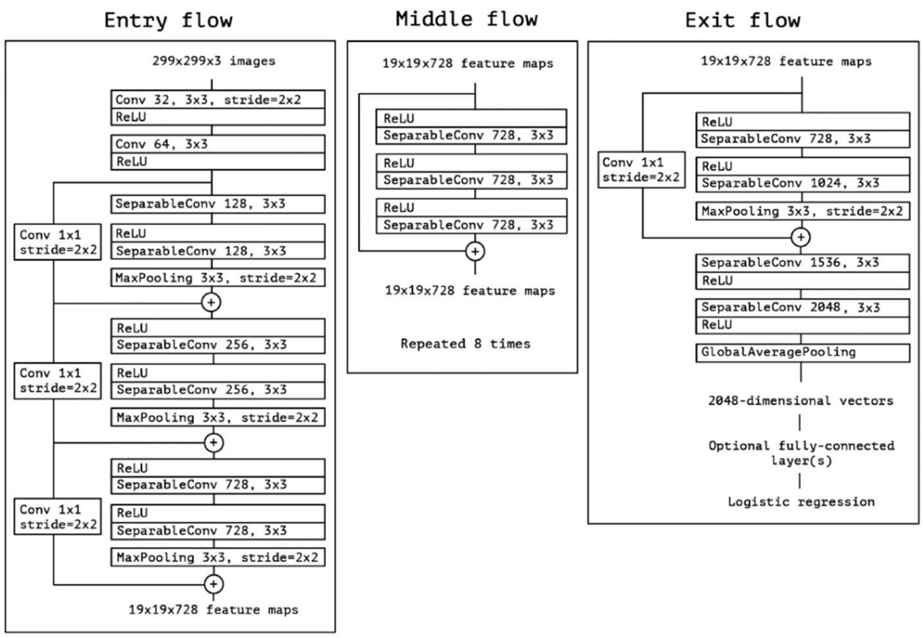

Fig. 11. Xception Architecture

Xception as feature extractor.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 10, 10, 2048)      20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dropout_1 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 12)                12300

=================================================================
Total params: 22,971,956
Trainable params: 22,917,428
Non-trainable params: 54,528
_________________________________________________________________
```

Fig. 12. Summary of layers in Xception as feature extractor model.

The Xception model, as illustrated in Fig. 12, utilizes pre-trained networks that have been trained on a large dataset and loaded with weights as the foundation of its architecture. To ensure that the last layer of the pre-trained model aligns with the classification task, a new classification layer is added on top of it. The final layer of the pre-trained model is substituted with a global average pooling layer that produces a feature map that is subsequently fed into the fully connected classifier. The fully connected layer is modified to suit the specific classification problem, featuring two layers containing 1024 units and 12 units, respectively, with relu and softmax activation functions, and some dropouts in between.

3.6 Compiling and training the model

Once the model is constructed, it is executed with the training and validation datasets. The training process is carried out for 100 epochs, and early stopping is employed to keep track of the validation accuracy and prevent overfitting. The Adam optimizer and categorical cross-entropy loss function are used, and the batch size is set to 16. Once the model is compiled and trained, it is evaluated.

4 Experimental results and discussion

Each of the four proposed CNN architectures for plant and weed seedling classification is evaluated and their performance is recorded using various metrics such as precision, confusion matrix, F1-score, recall, and accuracy.

Confusion Matrix: A confusion matrix, represented as C, is an n*n matrix where 'n' represents the number of classes. Each element in the i^{th} row (representing the actual class) and j^{th} column (representing the predicted class) of the matrix, denoted as $C_{i,j}$ shows the count of input samples that belong to class i and were predicted as class j.

For our classification problem, the confusion matrix has six rows and six columns

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	16	0	0	0	1	0	9	0	0	0	0	0
Charlock	0	35	3	0	0	0	0	0	0	0	1	0
Cleavers	0	0	25	1	1	0	0	0	0	0	1	0
Common Chickweed	0	0	0	56	0	0	0	0	2	3	0	0
Common wheat	0	0	0	0	22	0	0	0	0	0	0	0
Fat Hen	0	0	0	0	0	46	0	0	0	0	0	1
Loose Silky-bent	4	0	0	0	0	0	61	0	0	0	0	0
Maize	0	0	0	0	0	1	0	21	0	0	0	0
Scentless Mayweed	0	0	0	1	1	0	0	0	49	0	0	0
Shepherds Purse	0	0	0	0	0	0	0	0	2	21	0	0
Small-flowered Cranesbill	0	0	0	0	0	1	0	0	0	1	47	0
Sugar beet	0	1	0	0	1	0	1	0	0	0	0	35

(i) Confusion Matrix of self-built CNN model with 224x224 pixel size

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	18	0	0	0	1	0	7	0	0	0	0	0
Charlock	0	38	0	0	0	0	0	0	0	0	1	0
Cleavers	0	0	25	2	0	0	0	0	0	0	1	0
Common Chickweed	0	0	0	56	0	1	0	0	2	0	2	0
Common wheat	0	0	0	0	21	1	0	0	0	0	0	0
Fat Hen	0	0	1	4	0	42	0	0	0	0	0	0
Loose Silky-bent	9	0	0	0	1	1	54	0	0	0	0	0
Maize	0	0	0	0	1	0	0	20	0	0	0	1
Scentless Mayweed	0	0	0	1	0	1	1	0	45	3	0	0
Shepherds Purse	0	0	0	0	0	0	0	0	5	17	1	0
Small-flowered Cranesbill	0	0	0	0	0	0	0	0	2	0	47	0
Sugar beet	0	0	0	0	1	0	0	1	1	0	0	35

(ii) Confusion Matrix of feature extractor ResNet-50 with 224x224 pixel size

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	15	0	0	0	0	0	11	0	0	0	0	0
Charlock	0	39	0	0	0	0	0	0	0	0	0	0
Cleavers	0	1	25	0	1	0	0	0	1	0	0	0
Common Chickweed	0	0	0	59	0	0	0	0	2	0	0	0
Common wheat	1	0	0	0	20	0	1	0	0	0	0	0
Fat Hen	0	0	0	2	0	42	1	0	0	0	0	2
Loose Silky-bent	10	0	0	0	0	1	54	0	0	0	0	0
Maize	0	0	0	0	1	0	1	19	1	0	0	0
Scentless Mayweed	0	0	0	1	0	0	0	0	48	2	0	0
Shepherds Purse	0	0	0	2	0	0	0	0	6	14	1	0
Small-flowered Cranesbill	0	0	0	1	0	0	0	0	1	0	47	0
Sugar beet	1	0	0	0	2	0	0	1	0	0	0	34

(iii) Confusion Matrix of feature extractor Resnet-50 with 300x300 pixel size

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	20	0	0	0	0	0	6	0	0	0	0	0
Charlock	0	39	0	0	0	0	0	0	0	0	0	0
Cleavers	0	0	26	1	0	0	0	0	1	0	0	0
Common Chickweed	0	0	0	60	0	0	0	0	0	1	0	0
Common wheat	2	0	0	1	19	0	0	0	0	0	0	0
Fat Hen	0	0	0	1	0	45	0	0	0	0	0	1
Loose Silky-bent	5	0	0	0	0	0	60	0	0	0	0	0
Maize	1	0	0	0	0	0	0	21	0	0	0	0
Scentless Mayweed	0	0	1	2	1	0	0	0	47	0	0	0
Shepherds Purse	0	0	0	0	0	0	0	0	3	20	0	0
Small-flowered Cranesbill	0	0	0	0	0	0	0	0	0	0	48	1
Sugar beet	0	1	0	0	0	0	1	0	0	0	0	36

(iv) Confusion Matrix of finely tuned ResNet-50 (Last 15 layers trainable) + three dense layers with 224x224 pixel size

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	20	0	0	0	0	0	6	0	0	0	0	0
Charlock	0	38	0	0	0	0	0	0	0	0	0	1
Cleavers	0	1	25	0	1	0	0	0	0	1	0	0
Common Chickweed	0	0	0	60	0	0	0	0	1	0	0	0
Common wheat	0	0	0	0	22	0	0	0	0	0	0	0
Fat Hen	0	0	0	1	0	45	0	0	0	0	0	1
Loose Silky-bent	2	0	0	0	0	0	63	0	0	0	0	0
Maize	1	0	0	0	0	0	0	20	0	0	0	1
Scentless Mayweed	0	0	0	0	0	0	0	0	50	1	0	0
Shepherds Purse	0	0	0	0	0	0	0	0	3	20	0	0
Small-flowered Cranesbill	0	0	0	0	0	0	0	0	1	0	48	0
Sugar beet	0	1	0	0	0	0	1	0	0	0	0	36

(v) Confusion Matrix of finely tuned ResNet-50 (Last 15 layers trainable) + three dense layers with 300x300 pixel size

	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	24	0	0	0	0	0	2	0	0	0	0	0
Charlock	0	39	0	0	0	0	0	0	0	0	0	0
Cleavers	0	0	26	1	1	0	0	0	0	0	0	0
Common Chickweed	0	0	0	60	0	0	0	0	1	0	0	0
Common wheat	0	0	0	0	22	0	0	0	0	0	0	0
Fat Hen	0	0	0	1	0	46	0	0	0	0	0	0
Loose Silky-bent	2	0	0	0	0	0	63	0	0	0	0	0
Maize	1	0	0	0	0	0	0	21	0	0	0	0
Scentless Mayweed	0	0	0	0	0	0	0	0	51	0	0	0
Shepherds Purse	0	0	0	0	0	0	0	0	2	21	0	0
Small-flowered Cranesbill	0	0	0	0	0	0	0	0	0	0	49	0
Sugar beet	0	0	0	0	1	0	0	0	0	0	0	37

(vi) Confusion Matrix of Xception with 300x300 pixel size

Fig. 13. (i) – (vi) denotes Confusion matrices of different proposed models calculated on validation dataset

From the confusion matrices of the self-built CNN and ResNet-50 models, we can see that black-grass is more often misclassified as silky-bent and silky-bent as black-grass. Some of the Shepherd's Purse were misclassified as scentless mayweed, and there were very small misclassifications among other classes. Most of these misclassifications are removed by the Xception model.

$$Precision = (TP / (TP+FP)) \quad (1)$$

$$Recall = (TP / (TP+FN)) \quad (2)$$

$$\text{F1-score} = (2 \times ((\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}))) \quad (3)$$

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) \quad (4)$$

Classification Report

	precision	recall	f1-score	support
Black-grass	0.80	0.62	0.70	26
Charlock	0.97	0.90	0.93	39
Cleavers	0.89	0.89	0.89	28
Common Chickweed	0.97	0.92	0.94	61
Common wheat	0.85	1.00	0.92	22
Fat Hen	0.96	0.98	0.97	47
Loose Silky-bent	0.86	0.94	0.90	65
Maize	1.00	0.95	0.98	22
Scentless Mayweed	0.92	0.96	0.94	51
Shepherds Purse	0.84	0.91	0.87	23
Small-flowered Cranesbill	0.96	0.96	0.96	49
Sugar beet	0.97	0.92	0.95	38
accuracy			0.92	471
macro avg	0.92	0.91	0.91	471
weighted avg	0.92	0.92	0.92	471

Fig. 14. Classification Report for self-built CNN model with 224x224 pixel size.

Classification Report

	precision	recall	f1-score	support
Black-grass	0.61	0.54	0.57	26
Charlock	0.95	0.97	0.96	39
Cleavers	0.89	0.86	0.87	28
Common Chickweed	0.85	0.98	0.91	61
Common wheat	1.00	0.91	0.95	22
Fat Hen	0.93	0.89	0.91	47
Loose Silky-bent	0.78	0.86	0.82	65
Maize	0.95	0.91	0.93	22
Scentless Mayweed	0.84	0.84	0.84	51
Shepherds Purse	0.94	0.74	0.83	23
Small-flowered Cranesbill	1.00	0.94	0.97	49
Sugar beet	1.00	0.97	0.99	38
accuracy			0.89	471
macro avg	0.90	0.87	0.88	471
weighted avg	0.89	0.89	0.88	471

Fig. 15. Classification Report for feature extractor ResNet-50 with 224x224 pixel size.

Classification Report

	precision	recall	f1-score	support
Black-grass	0.56	0.58	0.57	26
Charlock	0.97	1.00	0.99	39
Cleavers	1.00	0.89	0.94	28
Common Chickweed	0.91	0.97	0.94	61
Common wheat	0.83	0.91	0.87	22
Fat Hen	0.98	0.89	0.93	47
Loose Silky-bent	0.79	0.83	0.81	65
Maize	0.95	0.86	0.90	22
Scentless Mayweed	0.81	0.94	0.87	51
Shepherds Purse	0.88	0.61	0.72	23
Small-flowered Cranesbill	0.98	0.96	0.97	49
Sugar beet	0.94	0.89	0.92	38
accuracy			0.88	471
macro avg	0.88	0.86	0.87	471
weighted avg	0.89	0.88	0.88	471

Fig. 16. Classification Report for feature extractor ResNet-50 with 300x300 pixel size.

Classification Report

	precision	recall	f1-score	support
Black-grass	0.71	0.77	0.74	26
Charlock	0.97	1.00	0.99	39
Cleavers	0.96	0.93	0.95	28
Common Chickweed	0.92	0.98	0.95	61
Common wheat	0.95	0.86	0.90	22
Fat Hen	1.00	0.96	0.98	47
Loose Silky-bent	0.90	0.92	0.91	65
Maize	1.00	0.95	0.98	22
Scentless Mayweed	0.92	0.92	0.92	51
Shepherds Purse	0.95	0.87	0.91	23
Small-flowered Cranesbill	1.00	0.98	0.99	49
Sugar beet	0.95	0.95	0.95	38
accuracy			0.94	471
macro avg	0.94	0.92	0.93	471
weighted avg	0.94	0.94	0.94	471

Fig. 17. Classification Report for Finely tuned ResNet-50(Last 15 layers trainable)+ three dense layers with 224x224 pixel size.

Classification Report

	precision	recall	f1-score	support
Black-grass	0.87	0.77	0.82	26
Charlock	0.95	0.97	0.96	39
Cleavers	1.00	0.89	0.94	28
Common Chickweed	0.98	0.98	0.98	61
Common wheat	0.96	1.00	0.98	22
Fat Hen	1.00	0.96	0.98	47
Loose Silky-bent	0.90	0.97	0.93	65
Maize	1.00	0.91	0.95	22
Scentless Mayweed	0.91	0.98	0.94	51
Shepherds Purse	0.91	0.87	0.89	23
Small-flowered Cranesbill	1.00	0.98	0.99	49
Sugar beet	0.92	0.95	0.94	38
accuracy			0.95	471
macro avg	0.95	0.94	0.94	471
weighted avg	0.95	0.95	0.95	471

Fig. 18. Classification Report for Finely tuned ResNet-50(Last 15 layers trainable) + three dense layers with 300x300 pixel size.

```
Classification Report
                           precision    recall  f1-score   support

              Black-grass       0.89      0.92      0.91        26
                 Charlock       1.00      1.00      1.00        39
                 Cleavers       1.00      0.93      0.96        28
         Common Chickweed       0.97      0.98      0.98        61
             Common wheat       0.92      1.00      0.96        22
                  Fat Hen       1.00      0.98      0.99        47
         Loose Silky-bent       0.97      0.97      0.97        65
                    Maize       1.00      0.95      0.98        22
        Scentless Mayweed       0.94      1.00      0.97        51
          Shepherds Purse       1.00      0.91      0.95        23
Small-flowered Cranesbill       1.00      1.00      1.00        49
               Sugar beet       1.00      0.97      0.99        38

                 accuracy                           0.97       471
                macro avg       0.97      0.97      0.97       471
             weighted avg       0.98      0.97      0.97       471
```

Fig. 19. Classification report for Xception with 300x300 pixel size

Table 2. The weighted average results based on the classification report (which is measured on validation set) of all proposed models

Models	**Pixel size**	**Weighted Average**		
		Precision	**Recall**	**F1-score**
Self-built CNN model	224x224	92%	92%	92%
ResNet-50 (as feature extractor)	224x224	89%	89%	88%
ResNet-50 (as feature extractor)	300x300	89%	88%	88%
ResNet-50 (finely tuned)	224x224	94%	94%	94%
ResNet-50 (finely tuned)	300x300	95%	95%	95%
Modified Xception (as feature extractor)	300x300	98%	97%	97%

Table 2. displays the averages for all classes. According to the results, the Xception with a 300x300 input pixel size produces the best results with precision of 98%, a recall of 97%, and an F1-score of 97%. According to the Xception Classification report, Charlock, Common Wheat, Small-flowered Cranesbill, and Scentless Mayweed have the highest recall values. And the Small-flowered Cranesbill and Charlock both received

the highest F1-score. The calculated Accuracy values based on eq (4) are mentioned in table 3.

Table 3. Accuracy of all proposed models.

Model	Pixel size	Accuracy
Self-built CNN model	224x224	92%
ResNet-50 (as feature extractor)	224x224	89%
ResNet-50 (as feature extractor)	300x300	88%
ResNet-50 (finely tuned)	224x224	94%
ResNet-50 (finely tuned)	300x300	95%
Modified Xception (as feature extractor)	300x300	97%

The accuracy values for the various models utilizing various pixel sizes are shown in Table 3. In comparison to other models, Xception produces the best results when employed as a feature extractor. With an input size of 224x224, the accuracy of the CNN model is 92%. ResNet50 obtains 89% accuracy when used as a feature extractor with 224x224 pixels, but decreases to 88% accuracy with 300x300 pixels. At 224x224 pixels, a precisely adjusted ResNet-50 model achieves 94% accuracy, while at 300x300 pixels, it reaches 95% accuracy. With a 97% accuracy rating, Xception performs best among all used models for recognizing plant and weed seedlings while using a 300x300 pixel size.

In order to have a deeper understanding of how well Xception performs, we computed its loss curve, which is displayed in Fig. 20. The validation loss is seen to decrease as the number of epochs rises, and overfitting is not observed.

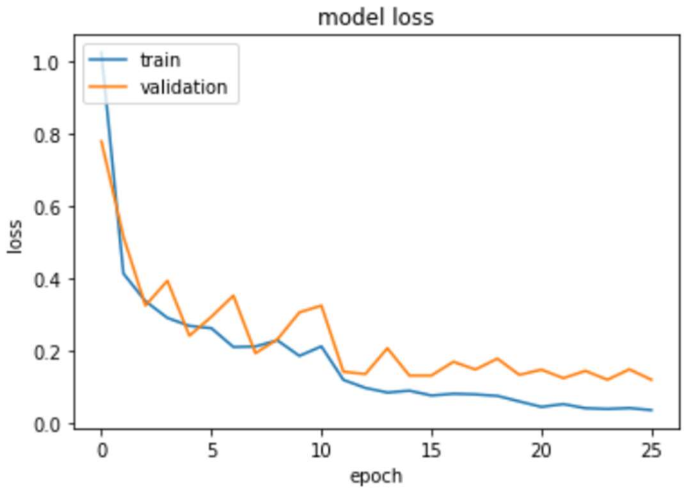

Fig. 20. Loss curve of Xception with 300x300 pixel size computed on training data and validation data

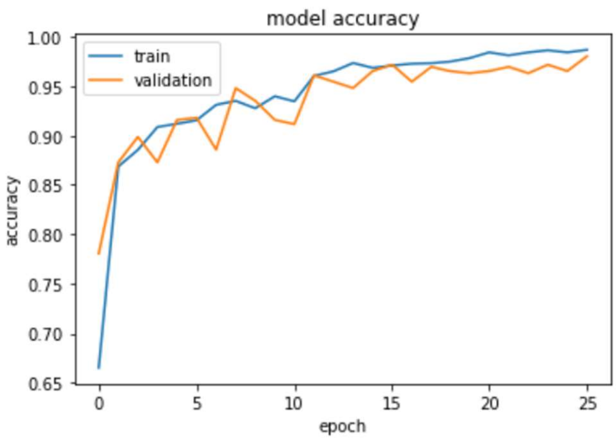

Fig. 21. Accuracy curve of Xception with 300x300 pixel size computed on training data and validation data.

From Fig. 21. it is observed that accuracy grows along with the number of epochs, and an overfitting problem is not observed.

5 Conclusion

Four CNN architectures are used in this work to accurately distinguish plants and weeds: self-built CNN, ResNet-50 as a feature extractor, finely tuned ResNet-50, and Xception as feature extractor models.

First, a self-built CNN model was employed for categorization, as demonstrated above, but it performed poorly. The ResNet model is then utilised to increase the accuracy; for 224x224 and 300x300 pixel sizes, respectively, the finely tuned Resnet-50 provided a better accuracy of 94% and 95%. Additionally, the Xception model is used to improve accuracy. The Xception model yields the highest accuracy. Farmers could use this model to automatically classify weeds and seedling

References

1. C. R. Alimboyong and A. A. Hernandez, "An Improved Deep Neural Network for Classification of Plant Seedling Images," 2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA), 2019, pp. 217-222, doi: 10.1109/CSPA.2019.8696009.Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).
2. M. Dyrmann, H. Karstoft, and H. Midtiby, "Plant species classification using deep convolutional neural network," Biosystems Engineering, 2016, pp. 72-80Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010).
3. A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny,J. Whinney et al., “Deepweeds: A multiclass weed species image dataset for deep learning,” Scientific reports, vol. 9, no. 1, pp. 1–12, 2019.
4. S. Murawwat, A. Qureshi, S. Ahmad, and Y. Shahid, “Weed Detection Using SVMs”, Eng. Technol. Appl. Sci. Res., vol. 8, no. 1, pp. 2412–2416, Feb. 2018.
5. S. L. Sanga, D. Machuve, and K. Jomanga, “Mobile-based Deep Learning Models for Banana Disease Detection”, Eng. Technol. Appl. Sci. Res., vol. 10, no. 3, pp. 5674–5677, Jun. 2020.
6. Y. A. LeCun, Y. Bengio, and G. E. Hinton, Deep learning, Nature, vol. 521, no. 7553, pp. 436444, 2015.
7. P. Kim, MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence. Apress, 2017.
8. L. D. Le Cun Jackel, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, B. Le Cun, J. Denker, and D. Henderson, Handwritten digit recognition with a back-propagation network, Adv. Neural Inf. Process. Syst., pp. 396404, 1990.
9. A. Krizhevsky, I. Sutskever, & G. E. Hinton, Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105), 2012.
10. C. Szegedy et al., Going deeper with convolutions, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 19
11. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, Int. Conf. Learn. Represent., pp. 114, 2015.

12. K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770778.
13. M. Dyrmann, H. Karstoft, and H. S. Midtiby, Plant species classification using deep convolutional neural network, Biosyst. Eng., vol. 151, no. 2005, pp. 7280, 2016.
14. O. Russakovsky et al., ImageNet Large Scale Visual Recognition Challenge, Int. J. Comput. Vis., vol. 115, no. 3, pp. 211252.
15. L. D. Le Cun Jackel, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, B. Le Cun, J. Denker, and D. Henderson, Handwritten digit recognition with a back-propagation network, Adv. Neural Inf. Process. Syst., pp. 396404, 1990.