Pertanika J. Sci. & Technol. 31 (1): 401 - 423 (2023)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Weed Detection in Soybean Crop Using Deep Neural Network

Vinayak Singh[1], Mahendra Kumar Gourisaria[1]*, Harshvardhan GM[1] and Tanupriya Choudhury[2]

[1]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, Odisha 751024, India
[2]Department of Informatics, School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India

ABSTRACT

The problematic and undesirable effects of weeds lead to degradation in the quality and productivity of yields. These unacceptable weeds are close competitors of crops as they constantly devour water, air, nutrients, and sunlight which are helpful for the maturation of crops. For better cultivation and good quality production of crops, weed detection at the appropriate time is an essential stride. In recent years, various state-of-the-art (SOTA) architectures were proposed to detect weeds among crop yields, but they lacked computational cost. This paper mainly focuses on proposing a customized state-of-the-art (SOTA) architecture and comparative study with transfer learning models for detecting and classifying weeds among soybean crops by concentrating on the low computational cost. The selected SoTA is beneficial for detecting weeds on a large scale with very low computational costs. In terms of selection, Maximum Validation Accuracy (MVA), Least Validation Cross-Entropy Loss (LVCEL), and Training Time (TT) were considered for proposing an objective function value system. In total, 15 proposed CNNs with 18 Transfer learning models were analyzed with the help of objective function value and various metric evaluations for finding the best and optimal architecture for weed classification. Experimentation and analysis resulted in C_{13} being robust and optimal architecture which outperformed every CNNs and Transfer learning model by achieving the highest accuracy of 0.9458 with an objective function value of 5.9335 and ROC-AUC of 0.9927 for the classification of weeds from soybean crops.

Keywords: Agriculture, ANN, CNN, deep learning, image recognition, machine learning, transfer learning

ARTICLE INFO

Article history:
Received: 22 January 2022
Accepted: 24 May 2022
Published: 09 November 2022

DOI: https://doi.org/10.47836/pjst.31.1.24

E-mail addresses:
vinayaksooryavanshi@gmail.com (Vinayak Singh)
mkgourisaria2010@gmail.com (Mahendra Kumar Gourisaria)
harrshvardhan@gmail.com (Harshvardhan GM)
tanupriya1986@gmail.com (Tanupriya Choudhury)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

© Universiti Putra Malaysia Press

INTRODUCTION

China, the United States of America (USA), Brazil, and India are major agricultural-producing countries worldwide. Agricultural activities play an important role in the world's economy, majorly for developing nations, as agriculture is the primary source of employment, income, and food. So, agriculture needs a major improvement for fruitful results. Farmers are willing to have less investment and high yielding of crops. So major problem faced by almost every farmer is weed detection problems which are unwanted crops that reduce the growth of crops.

Weed detection in yields is the major reason behind the high cost of crops. They have been a constant threat to the agricultural sector for decades as many techniques have failed to detect weeds at the right time. For the growth of crops, water, air, nutrients, and sunlight are major components, but weeds outgrow and consume a major portion of these which causes losses in crop production every year. Reduction in quality, quantity, and production rate with high prices are major effects of improper growth of crops with weeds.

Majorly this paper will focus on the Soybean crop, which is largely grown in Brazil and consumed by both Homo sapiens and Animalia. Soybean is a widely grown edible oilseed, and they are in the family of Fabaceae, and they also belong to Glycine and Plantae in terms of genus and kingdom, respectively. Animal consumes it through soybean meal, and human consumes it as oil. According to the stats collected by Soystats, soybean is a major crop, with 341.8 million Metric Tons of production in 2019. Brazil is the world's major soybean producer, sharing 126 million Metric Tons (Soystats, 2020). Soybean shares almost 25% of our edible oil in the global world. These salient features are likely to be highly rich in fibers, protein, phytoestrogens, low saturated fat, free from cholesterol and lactose, a source of antioxidants, and a good source of omega-3 fatty acids. Weeds are unwanted plants that grow with crops. They are a close competitor for nutrients, space, sunlight, and water with crops, which results in harvesting difficulties, decrement in the quality of crops, and crop diseases, due to which cost of production increases, the risk for pests degrade the rate of production, impurity with moisture in the grains and commercial value of cultivated land is decreased. Weed removal includes various processes like weed removal manually, a sprinkling of pesticides, and many more. Leaves of weeds can be classified into weeds and grasses, where weeds are broadleaf weeds grown between crops. About 40% of global harvesting losses have been caused due to weeds and their damage, besides pests.

Nowadays, farmers are facing problems in growing crops as the input costs are high, but the results of the production of crops are not up to a satisfactory level due to the unwanted growth of weeds. So, machine learning and deep learning will play a crucial role in properly classifying weeds from soybean crops, which will lead to excellent growth of crops, and the quality of crops will be better in terms of earlier stages where weeds were indulged for sharing nutrients. Manually handling weeds is difficult as it has high labor costs and is not an efficient method for detecting newly growing weeds. Due to weeds, farmers use herbicides that cause crop damage and may lead to various human diseases.

Many researchers have already contributed to this agricultural sector for weed detection as (Aravind et al., 2020) discussed four major crops where ten different crop diseases were included and models that were used pre-trained models like VGG-16, VGG- 19, and four more where they gained the highest accuracy of 97.3%. Classification and detection of soybean crops were done by (Ahmad et al., 2021) by using pre-trained VGG-16 for classification and YOLOv3 for localization and identification of weeds, where VGG 16 scored an accuracy of 98.80 and had an F1 score of 99%, whereas YOLOv3 had mAP score of 54.3%. All these implementations were also compared in terms of libraries, namely Keras and Pytorch. In 2017, (Ferreira et al., 2017) used ConvNets (CNNs) and proposed a Convolutional Neural Network Architecture for classifying weeds among soybean crops with an accuracy of 98% and distributed them into 4 classes.

Canny edge detection was used by (Badage, 2018) to detect plant-related diseases. The paper focused on two phases. The first process concentrated on training the model, and the second phase concentrated on monitoring the crop activity and finding whether they are healthy or not. Tang et al. (2017) classified weeds in soybean crops using Convolutional neural networks with an unsupervised technique, i.e., the K-mean feature as the pre-trained process, which achieved an accuracy of 92.89%.

Similarly, (Veeranampalayam et al., 2021) compared object detection using UAV imagery where a Single-shot detector and Faster RCNN were used for weed detection and had IoU of 0.85 for Faster RCNN and 0.84 for Single Shot Detection. Chen et al. (2018) used pre-trained deep learning models such as VGG16 and VGG 19 for detecting diseases in vegetables and fruits like cucumber, rice, wheat, grapes, and many more. The implementation was based on a total of 15000+ images, and the authors created the dataset. Finally, Etienne et al. (2019) proposed the site-specific Weed Management procedure, an automated weed detection system for four common types of weeds. Their proposed methodology consisted of four primary steps: a collection of multispectral and colored images from UAV-based sensors in soybean and cornfields, creation of normalized differential vegetation index (NDVI), image smoothing and thresholding to NDVI imagery, and drawing the bounding boxes with data labeling then the architecture was trained on the various growth stage of crops.

Maximum likelihood classification was used by Asad et al. (2020) to remove the background of images. Further, they labeled the data, and that labeled data was used for training semantic segmentation models and classifying crops. Various deep learning architectures like SegNet and UNet were compared, and ResNet50-based SegNet showed the best result. Similarly, deep convolutional neural networks like VGGNet and GoogleNet were used by Yu et al. (2019) for weed detection among turf grass, where VGGNet had the highest F1 score, which was above 0.95. On the other hand, DetectNet gained an F1 score of 0.99 for detecting weeds growing in dormant Bermuda grasses. Table 1 shows the tabular representation of five related works with their advantages and disadvantages.

Table 1
Tabular representation of various related work

Author Name and Year	Techniques	Advantage	Disadvantage
Aravind et al. (2020)	VGG-16, VGG19, DenseNet201, GoogleNet, ResNet101	Google Net was the best performer, with the highest accuracy of 97.3%. A good comparative study.	Only pre-trained architectures were compared. No proposed system or customized architectures were proposed.
Badage, (2018)	Canny Edge Detection	Proposed a phase-based system for plant disease detection system. Used Gaussian filter for removing noise from images. Pre-processing was good before training.	No classification process was included. No comparative study with other related works.
Ahmad et al., (2021)	Yolov3, VGG-16	Used VGG-16 for classification and Yolov3 for detection and identification of weeds. The results achieved were outstanding	No comparative study. No customized and proposed neural networks. Only pre-trained networks were considered for classification.
Tang et al. (2017)	CNNs, K-mean	The highest accuracy achieved was 92.89%. Used K-means with CNN architectures for training and pre-processing.	Lacked in comparison with other models and related works
Asad et al. (2020)	Maximum likelihood classification, VGGNet, SegNet	Maximum Likelihood classification was used for background removal. Pre-trained architectures were used for semantic segmentation	The results gained were not good. No comparison with other related works.

Most papers focused on detecting weeds, but in a few articles, the results were not satisfactory. Few lacked a comparison study of architectures, and most of them had used pre-trained architectures. However, this article is especially related to the cultivation of soybean and the classification of weeds from soybean. This paper emphasizes pre-trained not only models but also includes various state-of-the-art architectures. This paper broadly focuses on classifying weeds among soybean crops using Machine learning and deep learning techniques via customized and proposed CNN architectures and transfer learning models. Different configurations of proposed CNNs were compared, and optimal architecture was selected based on a maximum objective function focused on all three domains of maximum validation accuracy, least validation cross-entropy loss, and training time. Best CNN was selected and was compared with existing transfer learning models in terms of all metrics evaluation. In total, 18 different transfer learning models came into play, and 15 different parameterized CNN architectures were compared. The performance analysis consisted of all the domains of metrics evaluation, including objective function value and AUC- ROC.

This real-time deployment of architectures and transfer learning techniques will help detect weeds. Therefore, it can help farmers greatly by reducing the usage of herbicides and increasing soybean crop health and production of soybeans. The paper's next sections are Materials and Methods, Results, Conclusion and Future Direction. We evaluated all transfer learning models and ANN and CNN architectures to find the best technique and ideal model with maximum objective value function for the dataset of weed detection in soybean crops used in Section 3.

MATERIALS AND METHODS

This section will focus on the methodology used for classifying weeds from soybean. Weeds like Broadleaf and Grasses can be easily differentiated from our soybean crops, and the technology used for computational work and experimentation is specified in this section. This section is further divided into Dataset Collection, Convolutional Neural Network (CNN), Transfer Learning and Software and Hardware.

Dataset Collection

The dataset used consisted of 15,336 images which were collected from Mendeley data, and the dataset was deployed by Federico Peccia and Group, which captured all the images consisting of four people, namely Alessandro dos Santos Ferreira, Hemerson Pistori, Daniel Matte Freitas, and Gercina Gonçalves da Silva (Ferreira et al., 2017). These images were from Campo Grande, Mato Grosso do Sul, Brazil, from a soybean plantation area captured from the Phantom DJI 3 Professional drone. For our experimentation, we have removed the soil class from the dataset and converted it into a binary class with a total image of 12,087.

The modified dataset has been classified into two parts: soybean and weeds, which contain Grass and Broadleaf. A sample of images from the dataset for classification among both categories is shown in Figure 1. Table 2 shows the distribution of the dataset among three categories, i.e., Training, Testing, and Validation.

Table 2
Dataset distribution into three categories

	Soybean	Weeds
Training	4426	2827
Testing	1475	942
Validation	1475	942
Total	7376	4711

Convolutional Neural Network (CNN)

ConvNet (CNN) is a basic operational neural network that is a backbone in deep learning, contributing to diagnoses and recalling visual imagery. There is a wide range of CNN applications, such as classification

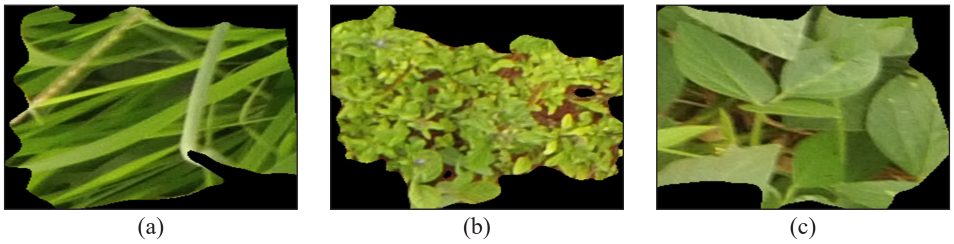


Figure 1. (a) Grasses (b) Broadleaf (c) Soybean leaf

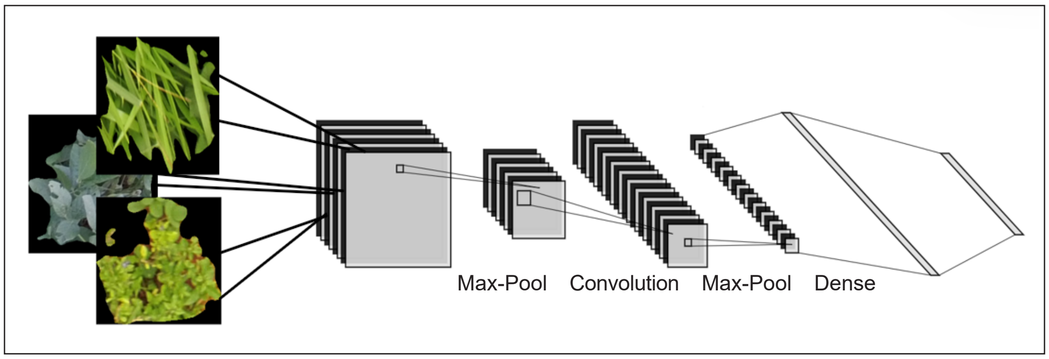


Figure 2. Convolutional Neural Networks (CNN)

of Brain Tumor (Singh et al., 2022a), tuberculosis detection and classification using CXR images (Al-Timemy et al., 2021; Singh et al., 2022b, retinal disease detection and classification (Rajagopalan et al., 2021; Sarah et al., 2021), face recognition (Khalajzadeh et al., 2014), and diagnosis and classification of skin cancer (Sannigrahi et al., 2021), sperm classification (Chandra et al., 2021), ductal carcinoma (GM et al., 2022) and arrhythmia detection (Alfaras et al., 2019; Gourisaria et al., 2021). These neural networks are highly interactive with imagery, where they assume and learn the patterns associated with the images. CNN frameworks are composed of 4 layers: Convolution layer, Max-pooling, Flattening, and Full Connection. Figure 2 shows the basic block diagram for CNN.

The convolutional layer is the initial and primary layer of the Convolutional neural network. This network follows the convolutional theorem, which states that a property of equivariant is equivariant to the translation operations. Mathematically, the convolutional theorem can be represented as Equation 1:

$$x(y(n)) = y(x(n)) \tag{1}$$

Where n-various input images, x is the convolutional operation, and y is the image translation operation.

The pooling layer is the second layer associated with CNNs, where they provide us the facility to reduce feature maps retrieved from the convolutional layer. These pooling layers play an important role while reducing the dimensions and extracting the major features from an input image. This layer provides the ability to down-sample the feature map via spatial variance. The pooling layer can be classified into two categories: Average-Pooling and Max-Pooling. In our case, Max-pooling was considered for the implementation of CNN architectures. Output dimension can be calculated by using Equation 2:

$$Dimension_{output} = \frac{(m - f + 1) \times (n - p + 1)}{q(q \times o)} \quad (2)$$

Where m represents height (feature map), n represents width (feature map), o represents channel (feature map), p represents the filter size (feature map), and q represents stride length (feature map).

After multi-dimension output retrieval, these features are passed through the flattening layer, which converts the output of the max-pooling layer into a 1-D array, an input for the last fully connected layer. Finally, various operations are performed on a 1-D array received from the flattening layer in a fully connected layer, which turns into output per requirements.

In terms of depth, if the neural network is shallow, then there might be a chance of under-fitting. So variation in the depth of the convolutional layer plays a vital role in proposing a better classification architecture. In terms of depth, CNN architectures get enriched in parameters, and deep neural networks are capable of training on multiclass and heavy datasets for better and good computational results. Subsequently, various optimizers such as softmax and sigmoid play an important role during the operation by efficiently optimizing the output and helping in categorization or classification. In our case, softmax was considered with the binary number of outputs.

So, mathematically we can say in Equation 3:

$$\underbrace{\mathrm{A} * \mathrm{A} * \mathrm{A} \ldots\ldots\ldots * \mathrm{A} * \mathrm{A}}_{\mathrm{n}} = \mathrm{A}^{\mathrm{n}} \quad (3)$$

Where A is a single convolutional layer with different parameters and n is the number of layers implemented by users.

Our approach proposed customized CNN architectures based on parameters like Input Sizes of Images, Pooling Size, Feature Detection, L1, L2, Dropout and Batch Normalization, Number of Convolution layers, Artificial layer, and Kernel Size. As a result, CNN architectures' parameters varied within the most suitable range for finding the best architecture with suitable parameters. Furthermore, regularization conditions were closely enclosed with heavy architectures, which prevented them from overfitting the dataset.

Transfer Learning

Transfer learning architectures are complex CNNs and highly enriched in parameters and layers. Some transfer learning models used for classifications are Inception, MobileNet, VGG, DenseNet, ResNet, EfficientNet, Xception, and NASNet. These models have come up with better accuracy when compared with normal ConvNet architectures. Initially, all these architectures are proposed in a yearly competition known as ImageNet, where all these architectures are trained in thousands of classes. The standard input image size is set to be (224, 224, 3) for RGB images and (224, 224, 1) for black and white images, respectively.

Majorly they are used as pre-trained models where these architectures are imported, and the last layer is trained according to the various dataset by the user. Transfer learning can be implemented as a heterogeneous method in various states, whether on text and image connections or performing computational matrix calculations. Transfer learning connects various fields and has various applications like text classification, text clustering, image and clustering, sentiment classification, and metric learning. Table 3 shows the configuration of all the used transfer learning models.

Table 3
Different models with size and parameters

Model	Size	Parameters (millions)
Xception	88 MB	22.91
VGG16	528 MB	138.35
VGG19	549 MB	143.66
ResNet50	98 MB	25.63
ResNet101	171 MB	44. 70
ResNet152	232 MB	60.41
ResNet50V2	98 MB	25.61
ResNet101V2	171 MB	44.67
ResNet152V2	232 MB	60.38
InceptionV3	92 MB	23.85
InceptionResNetV2	215 MB	55.87
MobileNet	16 MB	4,.25
MobileNetV2	14 MB	3.53
DenseNet121	33 MB	8.06
DenseNet169	57 MB	14.30
DenseNet201	80 MB	20.24
NASNetMobile	23 MB	5.32
NASNetLarge	343 MB	88.94

Domain and tasks can be used as terms to explain transfer learning mathematically.

Let domain X consists of $\text{Å} \rightarrow$ feature space & $\text{P(A)} \rightarrow$ Marginal Probability Distribution where

$$\text{A} = \{\text{a1}, \text{a2}, \text{a3} \dots \text{aN}\} \text{ € Å}$$

Let specific domain $\text{X} = \{\text{Å}, \text{P(A)}\}$. It has 2 parts, i. e., Y → Label Space and $\{\text{f: Å} \rightarrow \text{Y}\}$, i. e., an objective oredictive function

For new instances, a function f is used for predicting $\text{f}(\text{a})$, i. e., the corresponding label. Now in Equation 4:

$$\text{K} = \{\text{Y}, \text{f}(\text{a})\} \tag{4}$$

where K is the task that is learned for the pair, i.e. $\{\text{a}_\text{a} \text{ € A}\}$, $\{\text{y}_\text{a} \text{ € Y}\}$ is part of training data $X_s \rightarrow$ Source domain (given), $K_s \rightarrow$ Learning task, $X_t \rightarrow$ Target domain, and $K_t \rightarrow$ Learning task.

Where in Equation 5:

$$X_s \neq X_t$$
$$K_s \neq K_t \quad (5)$$

Where, target predictive function $f_t(.)$ in K_t will be able to improve its learning when transfer learning is implemented with the help of knowledge in X_s & K_s.

These architectures are quite heavy and can easily and efficiently handle large datasets. We have used them with an Adam optimizer to compare and find the most suitable architecture for weed detection among soybean leaves.

Software and Hardware

All the models trained during the experiment were completed using Python-based programming with Keras and Tensorflow libraries on the Jupyter notebooks. The hardware system was configured with i5 8th Generation and 16GB RAM.

RESULTS AND DISCUSSION

In this section, we provide every detail regarding performance analysis and evaluation and a full comparison between different models that are trained on the weed dataset, and we selected the best CNN architecture model for comparing it with various transfer learning models for finding an optimal architecture for detection of weeds from soybean leaves. We divide this section into Experiments and Analysis, Transfer Learning Models, and Result of Selected Architecture versus Transfer Learning.

Experiments and Analysis

This section provides detailed knowledge about the structures of all the CNN models. It helps us find the optimal CNN architecture based on various parameters, including maximum objective function value. Various architectures were trained on different regularization conditions. For implementation, we had different regularization parameters like L1, L2, Dropout and Batch Normalization, Number of Convolution layer, Artificial layer, Input Sizes of Images, Pooling Size, Feature Detection, and Kernel Size and selection of the best architecture was based on Objective Function Value (OFV) where it was calculated by using 3 parameters that were Maximum Validation Accuracy (MVA), Least Validation Cross-Entropy Loss (LVCEL) and Training Time (TT). Objective Function Value can be expressed as Equation 6:

$$\frac{\text{MVA}}{\text{LVCEL} + \text{TT}} = \text{Objective Function Value} \tag{6}$$

All the information regarding all the 15 different CNN and ANN architectures with their MVA, LVCEL, and TT (in seconds) are included in Table 5. Table 4 contains the meaning of all the abbreviations used in Table 5. Figure 3 tells us about the structure of each architecture based on LVCEL, MVA, and TT. Figure 4 plots the training accuracy, and Figure 5 plots the cross-entropy loss of each architecture per epoch. From Table 5, we observed that architectures 9 and 8 performed very poorly concerning MVA and LVCEL as they had the least objective function values of 0.36000 and 0.39843, respectively. It might happen due to L2 and Batch Normalization as these parameters are responsible for the reduction in training time, which resulted in low MVA, led to a fall in objective function value, and further restricted the model from building a good connection with the dataset.

From Table 5, we can develop a relationship between the input image size and training time where TT is directly proportional to input image size as we can notice that if the size of the image is (64, 64), it takes much less time when compared to (128,128) input size of images. Architecture 5, 6, and 13 were fed with (64 and 64), and the rest were on (128,128), which can be observed in Table 5. From Figure 3, training time drastically changed in architectures where (64 and 64) were used as input image sizes. Architecture 9 took 4864 seconds to train the model, but it achieved an accuracy of 0.6103 due to the implementation of the L2 regularization condition. Architecture 8, 9, 10, and 11 show that the models were not performing well as the MVA was only 0.6103. It was due to L1, L2,

Table 4
Abbreviations used in Table 5

Abbreviation	**Meaning**
CL	The number of CNN Layers in an architecture.
AL	The number of ANN Layers in an architecture.
L1	Level 1 regularizations
L2	Level 2 regularizations
BN	Batch Normalization
DO	Dropout
IS	Input Size of image (64x64 or 128x128)
FD	Feature detected in a convolutional layer, layer-wise.
KS	Kernel Sizes for each convolutional layer, layer-wise.
PS	Pooling Sizes followed by every convolutional layer are always assumed to be a square matrix.
TT	Time taken in seconds to train the model.
LVCEL	Least Validation Cross-Entropy Loss achieved during training.
MVA	Maximum Validation Accuracy was achieved during training.

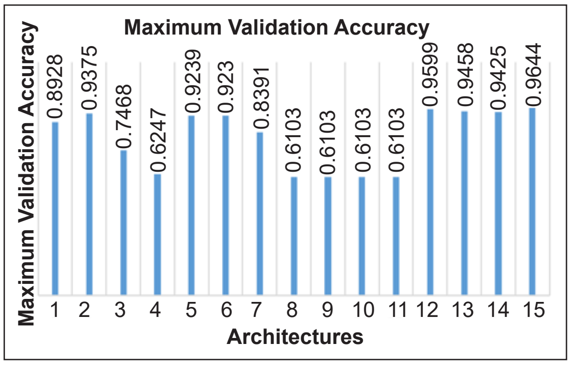
Table 5
Performance of 15 different CNN architectures on various parameters

S. No.	CL	AL	Regularizations				IS	FD	KS	PS	LVCEL	MVA	TT (in seconds)
			L1	L2	BN	DO							
1	2	3	✕	✕	✕	✕	(128,128)	{64,32}	{9,3}	{4,2}	0.2170	0.8928	3260
2	2	4	✕	✕	✕	✕	(128,128)	{64,32}	{9,3}	{4,2}	0.1808	0.9375	3756
3	2	4	✕	✕	✓	✓	(128,128)	{64,32}	{9,3}	{4,2}	0.5747	0.7468	4846
4	2	4	✕	✕	✓	✕	(128,128)	{64,32}	{9,3}	{4,2}	0.6097	0.6247	3791
5	2	4	✕	✕	✕	✓	(64,64)	{64,32}	{9,3}	{4,2}	0.2172	0.9239	960
6	2	2	✕	✕	✕	✕	(64,64)	{64,32}	{9,3}	{4,2}	0.2149	0.9230	1007
7	2	3	✕	✕	✓	✓	(128,128)	{64,32}	{9,3}	{4,2}	0.3601	0.8391	3611
8	3	4	✕	✕	✓	✕	(128,128)	{128,64,32}	{9,6,3}	{4,2,2}	0.6686	0.6103	4295
9	3	5	✕	✓	✕	✕	(128,128)	{128,64,32}	{9,6,3}	{4,2,2}	0.6687	0.6103	4864
10	4	4	✓	✕	✕	✓	(128,128)	{128,64,32,16}	{9,6,3,3}	{4,2,2,2}	0.6687	0.6103	1183
11	4	5	✓	✓	✕	✕	(128,128)	{128,64,32,16}	{9,6,3,3}	{4,2,2,2}	0.6693	0.6103	1439
12	4	5	✕	✕	✕	✕	(128,128)	{128,64,32,16}	{9,6,3,3}	{4,2,2,2}	0.1184	0.9599	1512
13	**4**	**5**	✕	✕	✕	✕	**(64,64)**	**{64,32,32,16}**	**{9,6,3,3}**	**{2,2,2,2}**	**0.1594**	**0.9458**	**503**
14	5	5	✕	✕	✕	✕	(128,128)	{128,64,64,32,16}	{9,6,6,3,3}	{2,2,2,2,2}	0.1645	0.9425	1592
15	5	6	✕	✕	✕	✕	(128,128)	{128,64,64,32,16}	{9,6,6,3,3}	{2,2,2,2,2}	0.1042	0.9644	1557

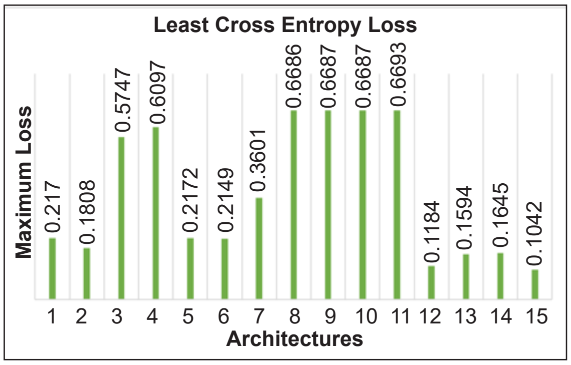
and BatchNorm conditions which led them to an overfitting condition.

However, the maximum objective function value of 5.93 was achieved by Architecture 13 (C_{13}). By this, we can conclude that simple architecture with a small input image performs better and is more stable and reliable. Various aspects such as Maximum Validation Accuracy (MVA), Least Validation Cross-Entropy Loss (LVCEL), Training Time (TT), and Objective Function Value (OFV) were considered for the selection of the best state-of-the-art architecture as the main motive for finding the best and optimal CNN architecture with the lowest computational cost for classification of weeds. For Further comparison and consideration, we have taken Architecture 13 as an ideal model to evaluate model performance and comparing with transfer learning models. Therefore, we will call architecture 13 C_{13} for comparing and further considering on weed detection dataset.

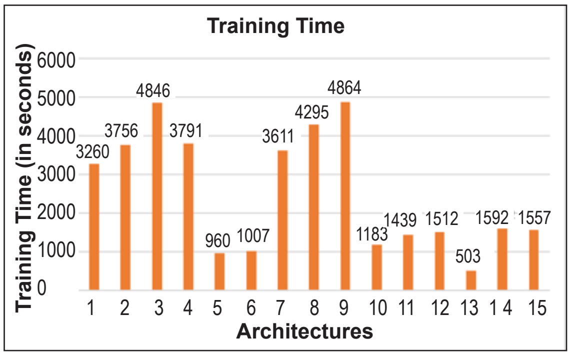
From Table 5, it was also observed that architecture fed with (64, 64) was providing better results when compared with (128, 128) fed architectures as they were less complex and computational power of convolutional neural networks were efficiently working with (64, 64) image size. Furthermore, it was noticed that all 3 architectures of (64, 64) gained accuracy above 0.90 and had less training time compared with (128,128) as the complexity increases as the image size increases. In our case, (64, 64) architectures performed very well as C_{13} was the best architecture when compared with



(a)



(b)



(c)

Figure 3. (a) Maximum Validation Accuracy (MVA) for each architecture. (b) Least Cross-Entropy Loss (LVCEL) for each architecture. (c) Training Time (TT) for each architecture

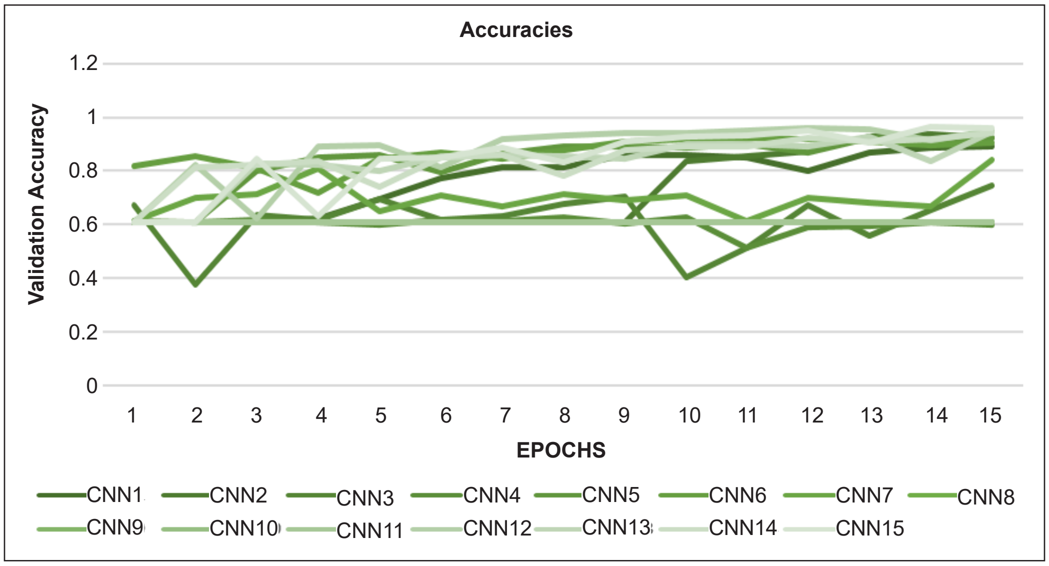


Figure 4. Validation accuracy training curve for each architecture per epoch

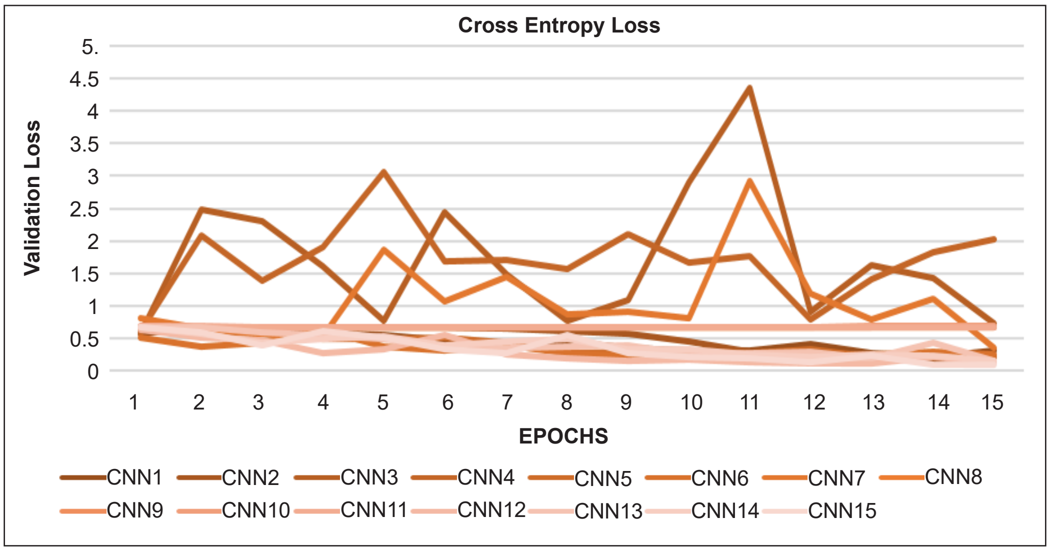


Figure 5. Validation cross-entropy loss training curve for each architecture per epoch

other models in terms of all the metrics where this architecture was balanced in terms of all three categories of maximum validation accuracy (MVA), least validation cross-entropy loss (LVCEL), and training time (TT). These three parameters led to the development of an objective function where our Architecture 13 outperformed all other CNN models.

Heavy architectures were considered with L1, L2, Batch Normalization, and Dropout to avoid the overfitting condition and the major concentration depended on proposing a low computational architecture for classification. Most architectures were trained on

128 × 128 image size. However, the objective function value was unsatisfactory as training time was greater than the time taken by architectures fed with 64 × 64 image size.

Transfer Learning Models

We have trained different transfer learning models on our dataset to compare our best CNN architecture and considered LVCEL, MVA, and TT for comparison and metric calculations. These architectures are pre-trained on large ImageNet data and heavily enriched parameters. Transfer learning architectures have dense layers of the convolutional network. All the information regarding MVA, LVCEL, and TT (in seconds) of the transfer learning model is provided in Table 6. Abbreviations used in Table 6 are given in Table 4. Figures 6, 7, and 8 discuss every transfer learning model's LVCEL, MVA, and TT.

From Table 6, we can notice that VGG-19 and NASNetLarge took 35800 and 30869 seconds, respectively, as they are highly rich in parameters, but they took approx. Nine hours of training which is very high. Xception performed pretty well as it achieved high accuracy but has taken less time compared to other architectures like VGG-16, VGG-19, and NASNetLarge. It performed well as a balanced model. MobileNet and MobileNetV2 are light architecture and performed well by achieving accuracy above 90%. The training time of these models was also less when compared to all other transfer learning models.

Table 6
Performance of various transfer learning models

S. No.	Transfer Learning Model	MVA	LVCEL	TT (in seconds)
1	Xception	0.9872	0.0698	11554
2	VGG-16	0.9888	0.0250	29989
3	VGG-19	0.9897	0.0325	35800
4	ResNet50	0.8540	0.4460	5924
5	ResNet101	0.8577	0.5327	15383
6	ResNet152	0.8531	0.6926	22447
7	ResNet50V2	0.8904	0.3341	8744
8	ResNet101V2	0.8635	0.3666	14092
9	ResNet152V2	0.8771	0.3301	21839
10	InceptionV3	0.8713	0.3547	6179
11	InceptionResNetV2	0.8577	0.3633	13421
12	DenseNet121	0.8093	0.4398	11063
13	DeneNet169	0.8395	0.4235	12217
14	DenseNet201	0.8448	0.4344	16676
15	MobileNet	0.9156	0.3418	3318
16	MobileNetV2	0.9090	0.3203	3279
17	NASNetLarge	0.7530	0.4860	30869
18	NASNetMobile	0.7815	0.5135	6564

They have taken only 3318 and 3279 seconds, respectively. NASNetLarge performed badly compared with other transfer learning architectures as the accuracy was only 75.30%, and its training time was 30869 seconds which was very high. Figure 8 shows that if transfer learning models have heavy parameters, they will take much more training time. The training time of transfer learning models was very high due to their high complexity compared with our state-of-the-art architectures.

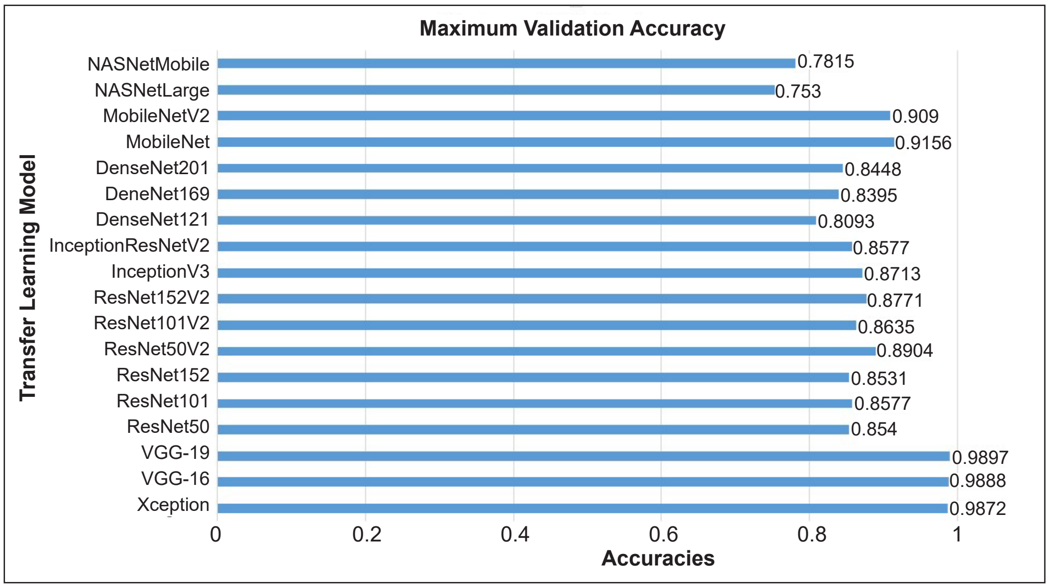


Figure 6. Maximum Validation Accuracy (MVA) for each transfer learning model

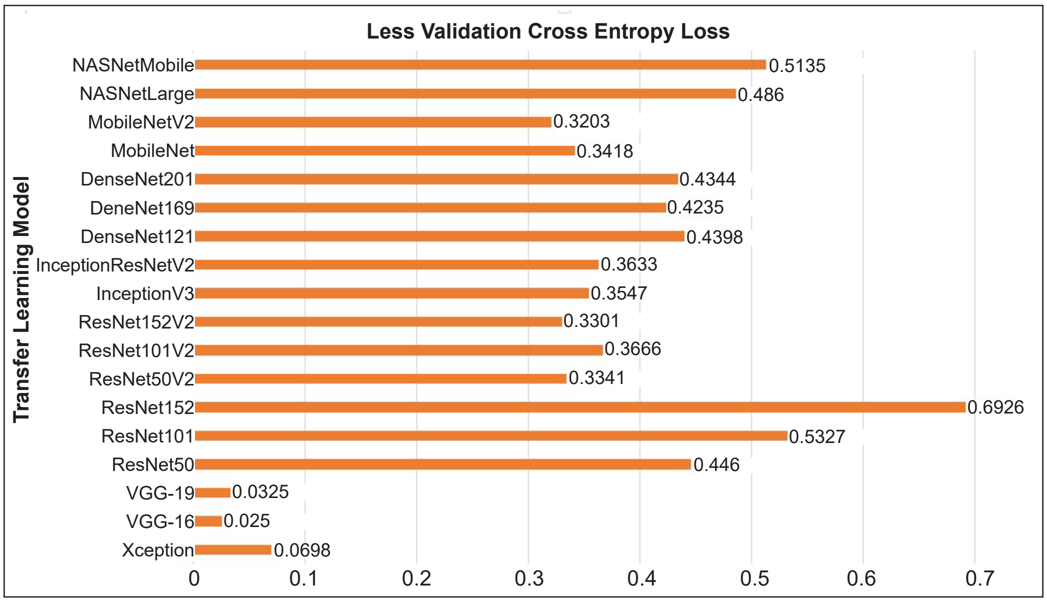


Figure 7. Least Cross-Entropy Loss (LVCEL) for each transfer learning model

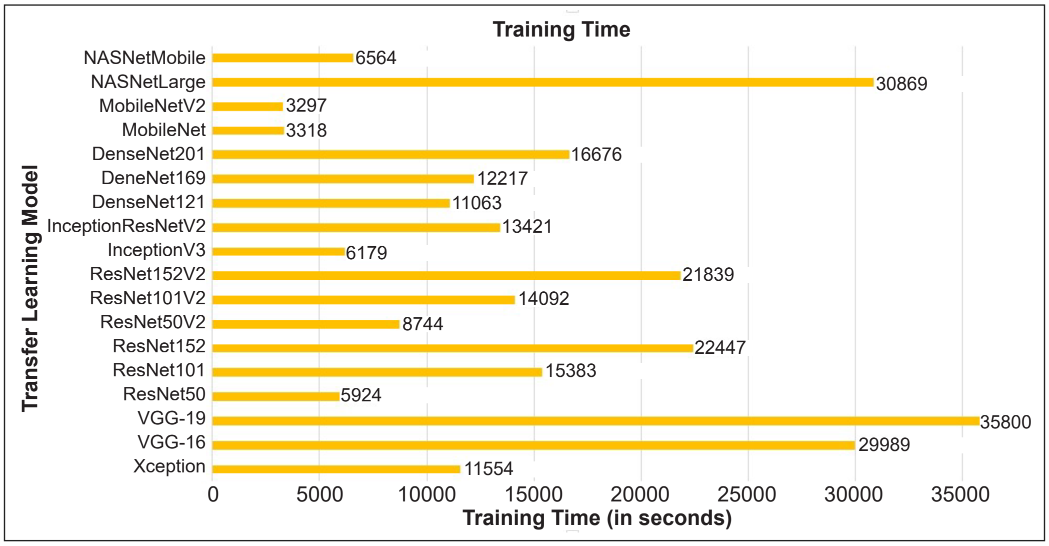


Figure 8. Training Time (TT) for each transfer learning model

While considering only the MVA, it was observed that 5 models gained MVA above 0.90, but when the time taken was considered, they had training time greater than CNN models as they were heavy in configuration, so training time was high. So, in terms of complexity and computational cost, these architectures were not much efficient when compared with models.

However, in the next section, we will compare all transfer learning models with our selected CNN architecture, i. e., C_{13} has achieved an objective function value of 5.93, and will judge all the models based on metric calculations performed in the next section.

Result of Selected Architecture versus Transfer Learning

In this section, our CNN architecture C_{13} will be compared with various transfer learning models based on various metric calculations. We have calculated all the metrics of the C_{13} and transfer learning models, which are mentioned in Table 7. Equations 7, 8, 9, 10, 11 and 12 represent the formula for the calculation of various performance metrics. Table 7 shows that Xception, VGG-16, and VGG-19 were strong competitors to C_{13} architecture. VGG-19 achieved the highest accuracy of 0.9897 and scored 0.9868 in MCC, which was the highest among all the models. Similarly, VGG-16 performed outstandingly in sensitivity and F1-score and achieved the highest metric calculation of 0.9978 and 0.9855, respectively. Xception was the best model in all the transfer learning architectures, and it also performed well in Specificity and Precision calculation, with a score of 0.9972 and 0.9958.

$$\text{Accuracy} = {(\text{TP} + \text{TN})}\Big/{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \qquad (7)$$

$$\text{Sensitivity} = {}^{\text{TP}}/_{(\text{TP} + \text{FN})} \tag{8}$$

$$\text{Specificity} = \text{TN} /(\text{FP} + \text{TN}) \tag{9}$$

$$Precision = \text{TP} /(\text{TP} + \text{FP}) \tag{10}$$

$$F1\,Score = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN}) \tag{11}$$

$$\text{MCC} = \frac{\text{TP x TN} - \text{FP x FN}}{\sqrt{((\text{TP}+\text{FP})*(\text{TP}+\text{FN})*(\text{TN}+\text{FP})*(\text{TN}+\text{FN}))}} \tag{12}$$

Further, if we notice NASNetMobile did not perform well and was not able to establish the relationship between dataset and model. So, NASNetMobile scored low in almost all the fields of metric calculations like Accuracy, Specificity Precision, F1 score, and MCC. The sensitivity of ResNet152 was not so good. According to the calculation, it was found to be only 0.7651, but in other fields, ResNet152 worked well in building the connection between the dataset and the model. Our C_{13} performed very well in all metric sections by achieving all the results above 89%.

From Table 7, we can see that Xception was the best in all the transfer learning models. Further, for more clarification, we plotted the AUC-ROC curve for all the architectures, calculated the Area under the Curve and Objective Function Value of all the transfer learning models, and compared it with our C_{13}. All details of the AUC score and OFV are mentioned in Table 8. Our architecture C_{13} outshined and gained a high AUC of 0.9927, as shown in Figure 8. Objective Function Value was calculated by comparing all the architectures using LVCEL, TT, and MVA, where C_{13} achieved an Objective Function Value of 5.9335, which is very high compared to all other transfer learning models. Xception performed very well and gained a value of 2.5788. VGG-16 and VGG-19 performed very well in all the domains. However, heavy parameters took much time for training, and this was the main reason behind the declination in objective function value, where the VGG-19 took almost 35800 seconds for training. Figure 9 shows the graphical comparison between C_{13} and all transfer learning architectures.

So, we can see that small architecture can perform well in less training time. Xception had a validation accuracy of 98% for every epoch, which was very good. Figure 10 shows all the ROC curve plotting where Xception, VGG-16, and VGG-19 were close competitors for C_{13} architecture, but our architecture had an AUC of 0.9927, which was better than other models. For the selection of the optimal architecture, all evaluation metrics were considered. However, majorly maximum objective function was considered as the paper mainly focuses on proposing a classification model of weed with a low computational cost.

Table 7
Metric calculation of selected CNN architecture and transfer learning models

Model Name	Accuracy	Sensitivity	Specificity	Precision	F1 Score	MCC	Confusion Matrix	
C_{13}	0.9458	0.9242	0.9642	0.9448	0.9344	0.8919	890	52
							73	1402
Xception	0.9872	0.9720	0.9972	0.9958	0.9837	0.9733	938	4
							27	1448
VGG-16	0.9888	0.9978	0.9833	0.9735	0.9855	0.9766	917	25
							2	1473
VGG-19	0.9897	0.9852	0.9925	0.9883	0.9783	0.9868	931	11
							14	1461
ResNet50	0.8540	0.9177	0.8277	0.6868	0.7857	0.6947	647	295
							58	1417
ResNet101	0.8577	0.9211	0.8313	0.6943	0.7918	0.7027	654	228
							56	1419
ResNet152	0.8531	0.7651	0.9275	0.8992	0.8267	0.7076	847	95
							260	1215
ResNet50V2	0.8904	0.9734	0.8555	0.7389	0.8401	0.7757	696	246
							19	1456
ResNet101V2	0.8635	0.9397	0.8327	0.6943	0.7985	0.7171	654	288
							42	1433
ResNet152V2	0.8771	0.9272	0.8544	07431	0.8250	0.7427	700	242
							55	1420
InceptionV3	0.8713	0.9376	0.8432	0.7176	0.8130	0.7324	676	266
							45	1430
InceptionResNetV2	0.8577	0.9359	0.8267	0.6815	0.7887	0.7049	642	300
							44	1431
DenseNet121	0.8093	0.8860	0.7826	0.5860	0.7054	0.5997	552	390
							71	1404
DenseNet169	0.8395	0.8991	0.8154	0.6624	0.7628	0.6629	624	318
							70	1405
DenseNet201	0.8448	0.9717	0.8029	0.6200	0.7570	0.6865	584	358
							17	1458
MobileNet	0.9156	0.9590	0.8940	0.8185	0.8832	0.8240	771	171
							33	1442
MobileNetV2	0.9090	0.8824	0.9260	0.8843	0.8834	0.8087	833	109
							111	1364
NASNetMobile	0.7530	0.9078	0.7202	0.4076	0.5626	0.4893	384	558
							39	1436
NASNetLarge	0.7815	0.8750	0.7539	0.5127	0.6466	0.5413	483	459
							69	1406

Table 8
AUC and Objective Function Value of selected CNN architecture and transfer learning models

Model Name	AUC	Objective Function Value
C_{13}	0.9927	5.9335
Xception	0.9887	2.5788
VGG-16	0.9860	1.1493
VGG-19	0.9894	0.9585
ResNet50	0.8237	1.4257
ResNet101	0.8281	0.8993
ResNet152	0.8614	0.6494
ResNet50V2	0.8629	1.5700
ResNet101V2	0.8328	1.1504
ResNet152V2	0.8529	0.9385
InceptionV3	0.8435	1.6928
InceptionResNetV2	0.8258	1.1776
DenseNet121	0.7689	1.0954
DenseNet169	0.8074	1.1126
DenseNet201	0.8042	0.9466
MobileNet	0.8980	2.1720
MobileNetV2	0.9045	2.2787
NASNetMobile	0.6906	1.1417
NASNetLarge	0.7329	0.5594

While considering all the parameters, Xception, VGG16, and VGG19 were ahead of our C_{13}, but when it comes to computational cost and training time, our C_{13} beat all the models. Our CNN model had less training time than transfer learning models. C_{13} gained high accuracy with minimal loss in less time, which resulted in proposing architecture for the classification of weeds among soybean with low computational cost.

Our selected C_{13} performed outstandingly in every metric calculation and had the highest objective function value of 5.9335, and the area under the curve was 0.9927, which is very high. C_{13} architecture had 251,730 parameters which were very less when compared to all transfer learning models. Testing of the model was done on 2417 images and where True Negative = 890, True Positive = 1402, False Negative = 52, and False Positive = 73 are shown in

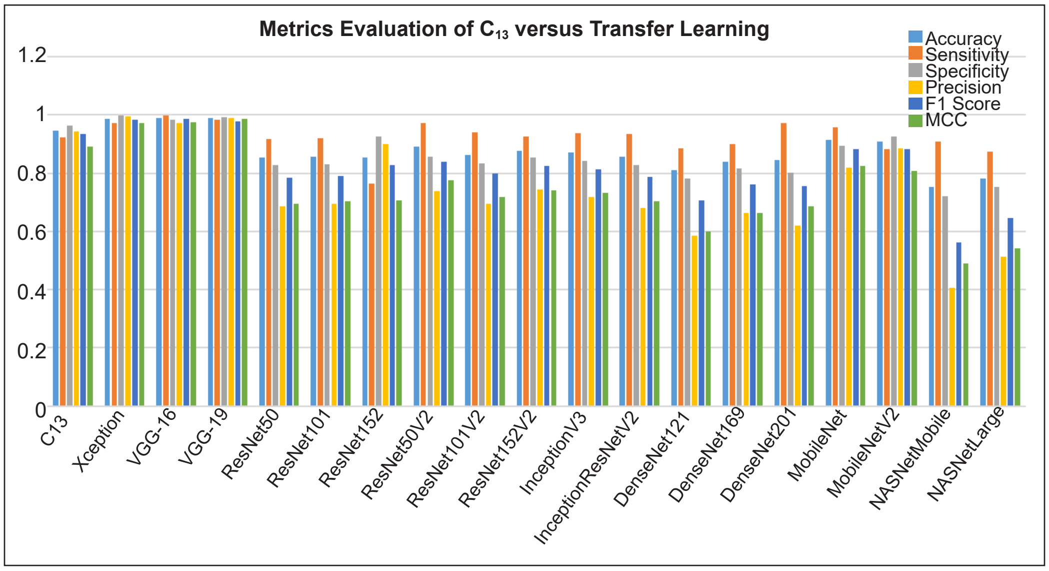


Figure 9. Graphical representation of metrics evaluation of C_{13} versus transfer learning

Figure 11. All the metrics were almost 90%, indicating our model was performing well and was excellent compared to all other transfer learning models in distinguishing between weeds and soybean leaves. These values are very high and far ahead of other proposed works in the related work section.

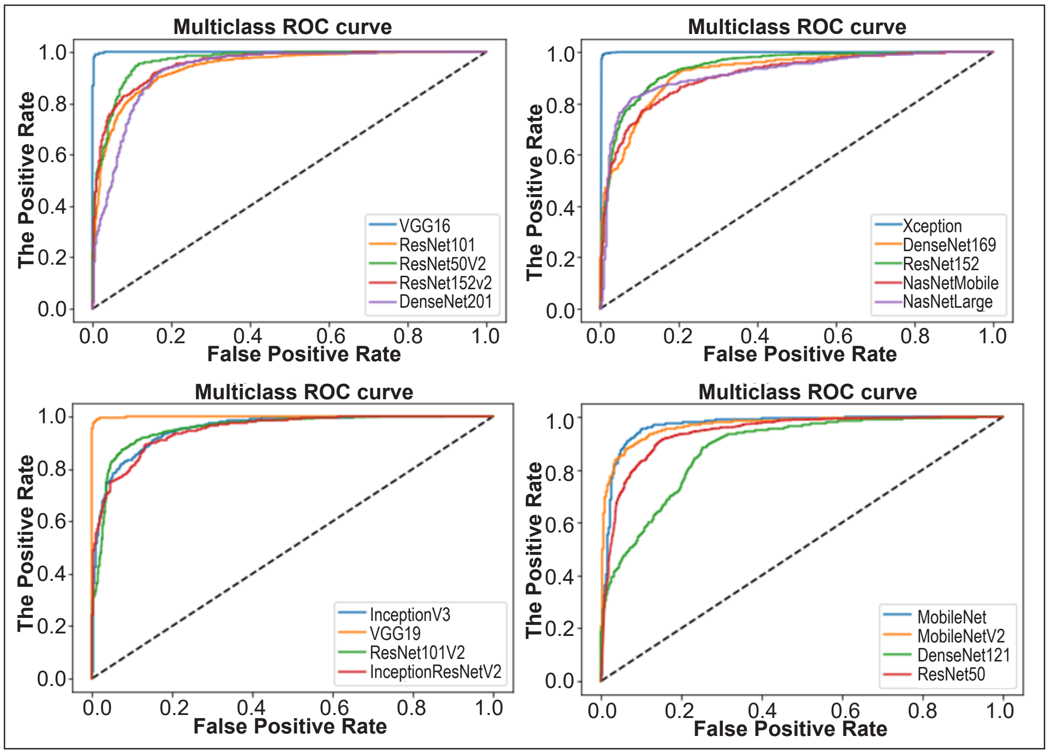


Figure 10. Receiver-operating characteristics (ROC) curve for all transfer learning models. The area under the curve is given in Table 7.

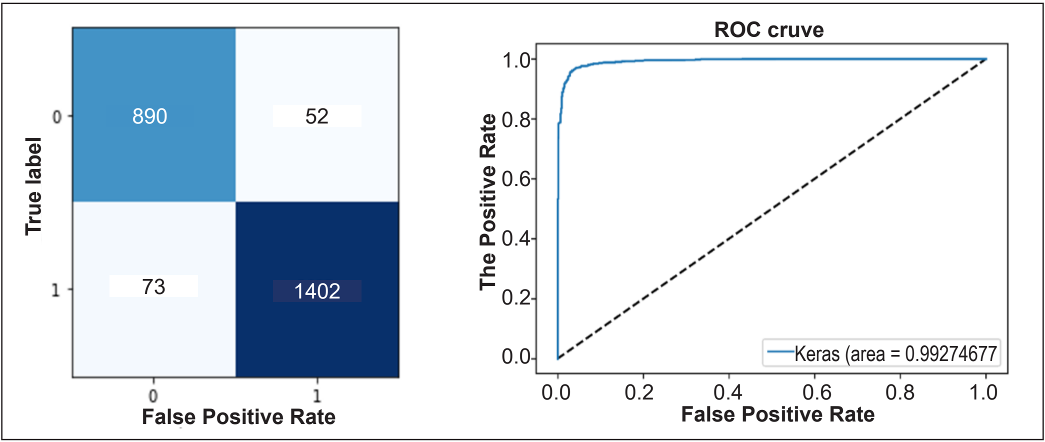


Figure 11. (L)The confusion matrix for C_{13} (see Table 7). (R**)** The receiver-operating characteristics (ROC) curve for C_{13} architecture. The area under the curve of C_{13} was 0.9927 units.

CONCLUSION AND FUTURE DIRECTION

Our research helps provide a detailed analysis of CNNs and transfer learning models and find an optimal architecture for detecting weeds from soybean crops. Based on Objective Function Value, we have selected the ideal model C_{13} for comparison with various transfer learning models in metrics evaluation which can easily reduce the computational cost. The best architecture was selected, and our proposed C_{13} outshines every domain as it has the least computational cost, which will help in easy training, faster detection rate, accurate classification, and reliability. As we compared our architecture with various architectures, we found that heavy parameterized transfer learning models like VGG-16 and VGG-19 have achieved very high accuracy. However, they lag in OFV calculation due to more training time, leading to a high computational cost. NASNetMobile performed poorly while building relationships with the dataset. So as learned from the findings in research work, we recommend that simpler architectures are good and reliable for learning, and they can perform much better than complex ones. We admit that C_{13} architecture performed well and can perform much better and gain a higher accuracy and stability by adjusting some hyper-parameters.

Our next goal is to propose another efficient model for classifying weeds from various crops. Some changes in the parameters of C_{13} can provide better results. More complex datasets can be used for weed detection and classification. The transfer learning technique, i.e., EfficientNet, can also be implemented to find a more stable and reliable model. As weeds have various categories, various complex models can be proposed for better detection. These techniques can work efficiently with few hyper-parameters tuning, leading to low computational cost as weed detection needs high accuracy. Our architecture had minimal computational cost as all the domains were considered for evaluating and detecting weeds among soybean leaves.

ACKNOWLEDGMENT

The author wishes to acknowledge no conflict of interest in publishing the paper. The author wants to thank Prof. Mahendra Kumar Gourisaria for always supporting and motivating nature. The author also acknowledges and thank Dr. Tanupriya Choudhury for his all-time support.

REFERENCES

Ahmad, A., Saraswat, D., Aggarwal, V., Etienne, A., & Hancock, B. (2021). Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems. *Computers and Electronics in Agriculture*, *184*, Article 106081. https://doi.org/10.1016/j.compag.2021.106081

Alfaras, M., Soriano, M. C., & Ortín, S. (2019). A fast machine learning model for ECG-based heartbeat classification and arrhythmia detection. *Frontiers in Physics, 7*, Article 103. https://doi.org/10.3389/fphy.2019.00103

Al-Timemy, A. H., Khushaba, R. N., Mosa, Z. M., & Escudero, J. (2021). An efficient mixture of deep and machine learning models for covid-19 and tuberculosis detection using x-ray images in resource-limited settings. In D. Oliva, S. A. Hassan & A. Mohamed (Eds.) *Artificial Intelligence for COVID-19* (Vol. 358, pp. 77-100). Springer. https://doi.org/10.1007/978-3-030-69744-0_6

Aravind, K. R., & Raja, P. (2020). Automated disease classification in (Selected) agricultural crops using transfer learning. *Automatika, 61*(2), 260-272. https://doi.org/10.1080/00051144.2020.1728911

Asad, M. H., & Bais, A. (2020). Weed detection in canola fields using maximum likelihood classification and deep convolutional neural network. *Information Processing in Agriculture*, *7*(4), 535-545. https://doi.org/10.1016/j.inpa.2019.12.002

Badage, A. (2018). Crop disease detection using machine learning: Indian agriculture. *International Research Journal of Engineering and Technology (IRJET), 5*(9), 866-869.

Chandra, S., Gourisaria, M. K., GM, H., Konar, D., Gao, X., Wang, T., & Xu, M. (2022). Prolificacy assessment of spermatozoan via state-of-the-art deep learning frameworks. *IEEE Access, 10*, 13715-13727. https://10.1109/ACCESS.2022.3146334

Chen, L., & Yuan, Y. (2018). Agricultural disease image dataset for disease identification based on machine learning. In J. Li, X. Meng, Y. Zhang, W. Cui & Du, Z. (Eds.), *International Conference on Big Scientific Data Management* (Vol. 11473, pp. 263-274). Springer. https://doi.org/10.1007/978-3-030-28061-1_26

Ferreira, A. D. S., Freitas, D. M., da Silva, G. G., Pistori, H., & Folhes, M. T. (2017). Weed detection in soybean crops using ConvNets. *Computers and Electronics in Agriculture, 143*, 314-324. https://doi.org/10.1016/j.compag.2017.10.027

Etienne, A., & Saraswat, D. (2019). Machine learning approaches to automate weed detection by UAV-based sensors. In *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV, International Society for Optics and Photonics* (Vol. 11008, Article 110080R). SPIE Digital Library. https://doi.org/10.1117/12.2520536

Gourisaria, M. K., Harshvardhan, G. M., Agrawal, R., Patra, S. S., Rautaray, S. S., & Pandey, M. (2021). Arrhythmia detection using deep belief network extracted features from ECG signals. *International Journal of E-Health and Medical Communications* (IJEHMC), *12*(6), 1-24. https://doi.org/10.4018/ijehmc.20211101.oa9

Harshvardhan, G. M., Sahu, A., Gourisaria, M. K., Singh, P. K., Hong, W. C., Singh, V., & Balabantaray, B. K. (2022). On the dynamics and feasibility of transferred inference for diagnosis of invasive ductal carcinoma: A perspective. *IEEE Access, 10*, 30870-30889. https://doi.org/10.1109/ACCESS.2022.3159700

Khalajzadeh, H., Mansouri, M., & Teshnehlab, M. (2014). Face recognition using a convolutional neural network and simple logistic classifier. In V. Snášel, P. Krömer, M. Köppen & G. Schaefer (Eds.), *Soft Computing in Industrial Applications* (Vol. 223, pp. 197-207). Springer. https://doi.org/10.1007/978-3-319-00930-8_18

Rajagopalan, N., Narasimhan, V., Vinjimoor, S. K., & Aiyer, J. (2021). Retracted article: Deep CNN framework for retinal disease diagnosis using optical coherence tomography images. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 7569-7580. https://doi.org/10.1007/s12652-020-02460-7

Sannigrahi, A., Singh, V., Gourisaria, M. K., & Srivastava, R. (2021). Diagnosis of skin cancer using feature engineering techniques. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)* (pp. 405-411). IEEE Publishing. https://doi.org/10.1109/ICAC3N53548.2021.9725420

Sarah, S., Singh, V., Gourisaria, M. K., & Singh, P. K. (2021). Retinal disease detection using CNN through optical coherence tomography images. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-7). IEEE Publishing. https://doi.org/10.1109/ISCON52037.2021.9702480

Singh, V., Gourisaria, M. K., GM, H., Rautaray, S. S., Pandey, M., Sahni, M., Leon-Castro, & Espinoza-Audelo, L. F. (2022a). Diagnosis of intracranial tumors via the selective CNN data modeling technique. *Applied Sciences*, *12*(6), Article 2900. https://doi.org/10.3390/app12062900

Singh, V., Gourisaria, M. K., GM, H., & Singh, V. (2022b). Mycobacterium tuberculosis detection using CNN ranking approach. In T. K. Gandhi, D. Konar, B. Sen & Sharma, K. (Eds.), *Advanced Computational Paradigms and Hybrid Intelligent Computing,* (Vol. 1373, pp. 583-596). Springer. https://doi.org/10.1007/978-981-16-4369-9_56

Soystats. (2020). *International: World soybean production*. The American Soybean Association. http://soystats.com/international-world-soybean-production/

Tang, J., Wang, D., Zhang, Z., He, L., Xin, J., & Xu, Y., (2017). Weed identification based on K-means feature learning combined with the convolutional neural network. *Computers and Electronics in Agriculture*, *135*, 63-70. https://doi.org/10.1016/j.compag.2017.01.001

Sivakumar, A. N. V., Li, J., Scott, S., Psota, E., Jhala, A. J., Luck, J. D., & Shi, Y. (2020). Comparison of object detection and patch-based classification deep learning models on mid-to late-season weed detection in UAV imagery. *Remote Sensing*, *12*(13), Article 2136. https://doi.org/10.3390/rs12132136

Yu, J., Sharpe, S. M., Schumann, A. W., & Boyd, N. S. (2019). Deep learning for image-based weed detection in turfgrass. *European Journal of Agronomy*, *104*, 78-84. https://doi.org/10.1016/j.eja.2019.01.004