An Efficient Weed Growth Rate Estimator

Yash Vishwakarma[1], Akhilesh A. Waoo[2]
{yashvishwakarma@hotmail.com[1], akhileshwaoo@gmail.com[2]}

AKS University, Satna, M.P., India[1], AKS University, Satna, M.P., India[2]

Abstract. This study drafts a new EfficientNetB0 CNN-based process of automatically classifying weed plants into different developmental phases. Images of weed plants growing within various crops across varying environmental constraints were used. About 90% of the images were used for training the proposed model. The performance of this EfficientNetB0 based convolutional neural network model was measured on a different additional set of 10% images never seen by the model. The model attained a very high 91% accuracy in identifying young single-leaf weed plants. Additionally, it attained an average accuracy of 73% in evaluating the count of leaves across all classes and an accuracy of 81% among all classes except one. The accuracy of the results conveys that this new method of using the EfficientNetB0 based model has a high potential to classify different developmental phases among distinct weed plants.

Keywords: computer vision; image classification; leaf counting; convolutional neural network; efficientnet

1 Introduction

Weed control persists to cause significant financial problems for arable farmers, and improved management division is necessary to check the escalating threat from herbicide resistance in weed. To help reduce costs for arable farmers, it is important to target weeds and develop strategic planning over an entire crop cycle. Proper weed control and balanced crop production can be achieved by optimally utilizing herbicides. This requires the knowledge of the conditions of weed by farmers in the farmland. It has been observed globally that there has been an under-exploited potential to acquire a 20%-40% reduction in the utilization of herbicides that can maintain weed control by targeting particular types of weeds. Yet, a recent study has shown that arable farmers are unenthusiastic to carry out farm surveys; moreover, identifying weed (during different developmental phases) is a significant challenge to keep a check on weed on fields.

Plant characteristics such as dimensions, count of leaves, and leaf appearance are influenced by numerous aspects, including (but not only) genetic and environmental factors (soil composition, moisture, nourishment, temperature, sunlight, and humidity). Also, fungal and insect invasions can change the dimensions and form of plant growth.

THEETAS 2022, April 16-17, Jabalpur, India

DOI 10.4108/eai.16-4-2022.2318148

Over the years, several industry norms have been employed to classify plants into distinct development phases based on the count of leaves and twigs. During the earlier phases of growth, the count of leaves is directly associated with the development phase; Thus, it's only probable to pick out the count of leaves and utilize this data to pinpoint the development phase of small weed plants. The development phase information can be connected with information on herbicide levels, which helps in doing adequate and efficient weed management [1]. Yet, estimating weed growth using non-detrimental and automated approaches like computer testing is a challenge that researchers still face [2]. A precision blend of computer vision and machine learning is a critical element in an automated weed management system. Recognizing a variety of grows and detecting leaf count with satisfactory accuracy is required [3].

To create a powerful model for determining and calculating leaves of weed plants, images in the dataset need to cover biodiversity concerning environmental conditions and plant growth categories. These conditions include visible sunlight settings, soil types, and plant types. A proper system should accurately identify the leaf count of weed plants before applying measures for weed control. A critical problem in calculating leaves is that the leaves of the weed plants often overgrow and could cover a crop or be covered by it, yet all the leaves have to be counted to get an idea of the growth phase and select the appropriate remedy for weed removal. Nevertheless, manual leaf counting in pictures can be problematic and time-consuming, even when done by professionals [4].

In recent years, deep convolutional neural networks (CNNs) have been a great hit among researchers working in computer vision and machine learning environments [5] due to their ability to produce functional features that distinguish images. CNNs have been widely used in the agriculture sector to solve problems involving plant species classification [6, 7], weed identification [8, 9, 10]. In contrast, the aim of this study was to devise an efficient way to calculate the value of leaves on weed plants in field photographs. The technique proposed in this research is an EfficientNetB0 based convolutional neural network trained on pictures of distinct weed plants at nine typical developmental phases.

2 Data Material

These photos were taken at various cropping seasons throughout the parts of Denmark, thus including a range of soil types, image resolutions, and lighting situations (Figure 1). An aspect that may impact how the convolutional neural network does leaf counting is if the leaves are stacked. Convolutional neural networks perform very well if the test examples are similar to the training examples; hence the training data should contain stacked or overlapping leaves. The dataset contained nine distinct classes, each numbered for the count of leaves. The dataset used in this research is available for public use at https://vision.eng.au.dk/leaf-counting-dataset/.

Fig. 1. A random selection of images from the dataset.

3 Methods

Neural networks consist of three layers, namely an input layer, one or more hidden layers with finite neurons, and an output layer. Because of these straightforward structures, there is a demand for specially developed feature extractor preceding the input layer. By distinction, convolutional neural networks (CNNs) are very deep and share weights between several neurons in a layer; in CNNs, the early layers extract low-level features such as colors and edges while the deeper layers extract high-level features such as leaves and twigs. The capability to extract hundreds of thousands of features indicates that convolutional neural networks can understand the images in the dataset similar to as humans perceive the images; Because of this, classifying images with a CNN produces a substantially lower error rate than any other classifying approach [11]. The layers in a CNN generally include several convolutions and sub-sampling filters (which are used for feature extraction), followed by fully-connected layers. The CNN model receives RGB images as input in batches, which are pre-processed for data augmentation purposes.

3.1 Image Pre-processing

This research employs an EfficientNetB0 based CNN model for leaf counting in weed plants. CNNs are generally trained on a large dataset (containing a huge number of images) to learn to extract features (that can be generalized to different kinds of images) from the input data. Also, these huge number of images enable the neural network to regulate itself and prevent the model from overfitting [12]. Overfitting occurs when the model's weights conform exceptionally well on the training set but perform very severely on validation or test set; because of this, the network is unable to detect notable discriminative features within new images from the validation or the test set. Overfitting complicates the prediction process for the model, and the model fails to accurately identify the new images that did not exist in the training set. Various strategies are used to prevent the model from overfitting, including augmenting the amount of training data or adding 'dropout' (which is described in Section 3.2.4). The images in the dataset were split into 90% training set and 10% validation set.

Furthermore, the training set was increased by using horizontal-flip, vertical-flip, rotation, zoom, shear, and height shift. Together with this data augmentation, the training images were randomly shuffled before being sent to the network in batches for training.

3.2 Network Architecture

There are several pre-trained CNN architecture models that are used for image classification tasks, but they are pre-trained on a set of a huge number of images (such as ImageNet [13]), images that are very distinct from the images in the dataset used for this research. However, even though the weed images are very distinct from those in the ImageNet dataset, the model can apply general features learned from the ImageNet dataset to weed images with very few training steps by fine-tuning the model's weights learned from the ImageNet dataset on the weed dataset. These pre-learned weights (instead of beginning with randomly initialized weights) help the model to easily learn the general features of the dataset used for this research. Commonly used pre-trained models include AlexNet [14], InceptionV3 [15], ResNet [16] and VGG [17]. For this research, EfficientNetB0 [18] was selected, as the name suggests is very much efficient computationally and its scaled-up version EfficientNetB7 also achieved state of art results on the ImageNet dataset which is 84.4% top-1 accuracy. EfficientNetB0 is a mobile-sized architecture that has 11 million training parameters and uses compound scaling that uniformly scales width, depth, and resolution with a fixed ratio.

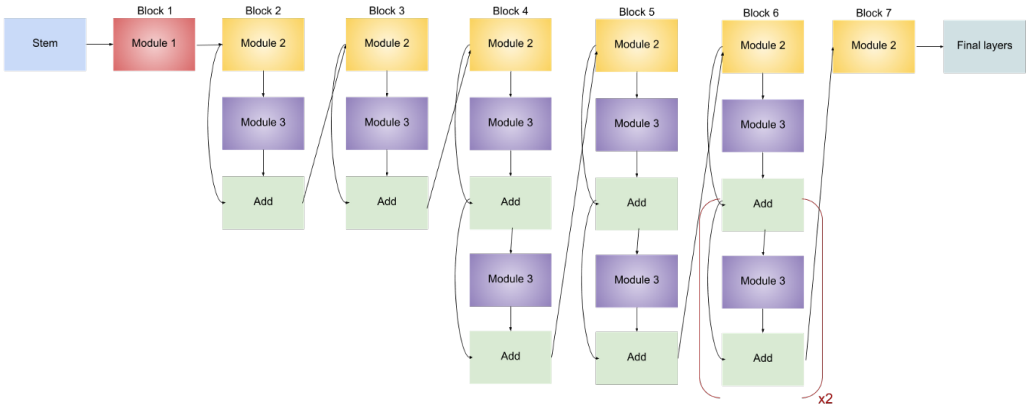

Fig. 2. EfficientNetB0 architecture.

3.2.1 Convolutional Layers

A convolutional layer is the primary building block of a convolutional neural network. This layer contains a set of filters (or kernels), parameters of which are to be learned throughout the

training. The feature extraction is achieved by convolving filters on the image to align with the input results in a map of activations known as a feature map. The convolution operation is shift-consistent, this means that the spatial connections between input pixels are maintained in the output feature map.

3.2.2 Activation Function

After every convolutional layer, a nonlinear activation function is used to help decide if the neuron would fire or not. The activation function executes a predetermined mathematical calculation on each value of its inputs, introducing non-linearities into the network. The pre-trained EfficientNetB0 used the Swish activation function [19], but the custom layer of our model used ReLU, and the final prediction layer used softmax for classification.

The ReLU activation function is defined as:

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{1}$$

The Softmax activation function is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}} \tag{2}$$

3.2.3 Global Average Pooling Layer

We add a global average pooling layer after the convolutional layer and feed its output straight into the softmax prediction layer. Global average pooling involves calculating the average for each patch of the feature map leading to the reduction of each channel to a single value. This leads to a reduction of the dimensions of the feature maps, which means the number of parameters to be learned and the computation required is decreased.

3.2.4 Dropout

Dropout is a regularization method, which prevents the neural network from overfitting [20]. Dropout is a process in which randomly-selected neurons in the network are disabled during the training stage, which adds noise and forces neurons within a layer to take more or less accountability for the inputs. The network becomes thinner because the outputs of a layer undergoing dropout are random.

3.3 Fine-Tuning the Network

EfficientNetB0 is mobile-sized architecture, has 11 million training parameters, and uses compound scaling that uniformly scales width, depth, and resolution with a fixed ratio. The network used RGB images as the input, and the images were labeled. The dataset was split randomly into two categories, training set comprising 90% of the dataset images, and validation comprising 10% images of the dataset images. During the training process, a dense layer followed the average pooling layer, and a 50% dropout rate was applied before the prediction layer. The final layer is comprised of a softmax prediction layer that predicts a multinomial probability distribution, which scales the output between (0, 1). This computes the predicted class for each image.
The network's weights were pre-trained on the ImageNet dataset to take benefit of the general features. The training on weed images used a batch size of 64 images. Adam optimizer [21] was used to decrease the error of the network and accelerate the training process. The learning rate reduction function was used to make the model converge even more quickly. Also, the early stopping function was also used to restore the best weights of the model during the early stoppage. Adam optimizer provides computational efficiency and uses very little memory. Ultimately, the accuracy of the model prediction was assessed by comparison with the validation set prediction.

3.4 Implementation

This research was implemented on Ubuntu 20.04 running on AMD Ryzen 7 5800× CPU (8 Cores/16 Threads at 4.8 GHz), 32 GB of DDR4 RAM (3733 MHz, C14), NVIDIA RTX 3080 GPU (10GB VRAM). The image pre-processing and other deep learning methodologies were carried out using the Tensorflow 1.5.5 library running on Python 3.8.

3.5 Results and Discussion

The EfficientNetB0 architecture was trained on the training set to classify weed plants into nine distinct development phase classes. To get a more reasonable understanding of errors and in turn improve the precision of the predictions, the model was trained for 70 epochs with a random learning rate reduction to help the model converge faster. Early stopping function was also used to restore the best weights. This allowed our model to predict the growth phase of weed plants per image with more confidence and higher precision. Figure 3 illustrates the accuracy of the training and validation sets. The training was automatically stopped after 67 Epochs with the help of early stopping function and the best weights were restored. At this point, the validation accuracy remained constant at 73%.

Due to the class imbalance problem, the model had a validation accuracy of 73%. By removing the class with the low amount of training data an 82% of validation accuracy was achieved. The confusion matrices are shown in Figures 4 and 5. Numbers near the prediction diagonal on each of the confusion matrices denote that the errors made by the model are oftentimes near the true class values.

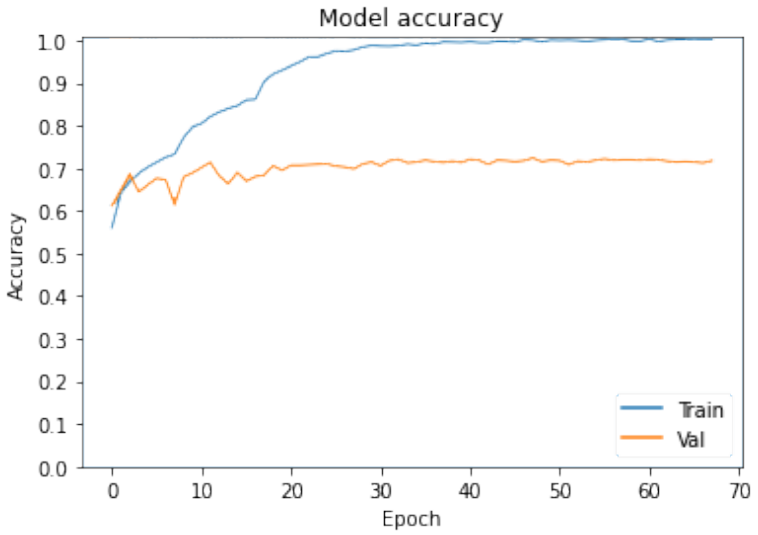

Fig. 3. Training and validation accuracy of the EfficientNetB0 model.

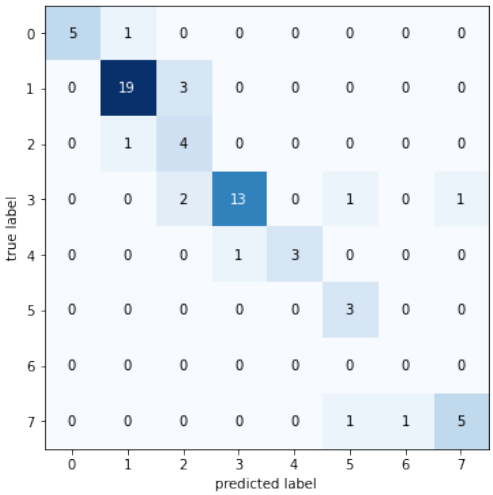

Fig. 4. Confusion matrix

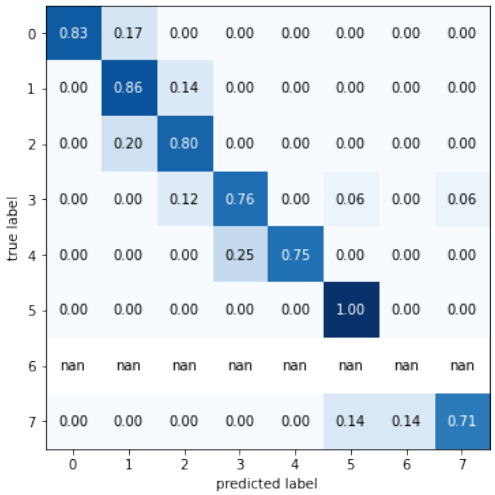

Fig. 5. Normalized confusion matrix

As the dataset suffers from a class imbalance problem, F1-Score was used as the accuracy metric as it takes into account how the data is distributed. F1-Score is defined as the harmonic mean of precision and recall. It is most commonly used for classification tasks that suffer from class imbalance

problem. The formula for F1-Score is as follows:

$$\frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3}$$

The various metrics class-wise are shown in Table 1. It is clear from the table that class 7 suffers due to the low amount of training data available for this class. Classes 1, 2, 4, 5, 6, 8 achieve an F1-score of 91%, 88%, 84%, 86%, 75% and 77% respectively. While classes 3 and 7 underperform due to the low amount of training data. The classes which had a good amount of training data performed generously on our model. Consequently, going by the results, it can be stated that the EfficentNetB0 model can be implemented with a 73% precision on field machinery. And if the weighted-average F1 score is considered, the accuracy of the model will be 83%. The weighted-averaged F1 score is computed by taking the mean of all per-class F1 scores while considering each class's support. Other scores such as micro, macro, and weighted scores are given in Table 2.

Table 1: Comprehensive results containing all the classes in the dataset.

Class (Count of Leaves	Precision	Recall	F1-score	Support
1	1.00	0.83	0.91	6
2	0.90	0.86	0.88	22
3	0.44	0.80	0.57	5
4	0.93	0.76	0.84	17
5	1.00	0.75	0.86	4
6	0.60	1.00	0.75	3
7	0.00	0.00	0.00	0
8	0.83	0.71	0.77	7
Accuracy			0.81	64
Macro average	0.71	0.72	0.70	64
Weighted average	0.87	0.81	0.83	64

3.6 Conclusion

This study presents an EfficientB0 based convolutional neural network model for assessing the development phases of leaves of distinct weed plants according to the count of leaves. Images in the dataset were collected in fields with a variety of conditions. The dataset had images in which plants overlaid on each other; the proposed model had to overcome this problem, which it did with good precision. The validation accuracy was 73%, whereas the network achieved a training accuracy of 99%.

Table 2: Macro, micro and weighted scores.

Score name	Scores
Accuracy:	0.81
Micro Precision:	0.81
Micro Recall:	0.81
Micro F1-score:	0.81
Macro Precision:	0.71
Macro Recall:	0.72
Macro F1-score:	0.70
Weighted Precision:	0.87
Weighted Recall:	0.81
Weighted F1-score:	0.83

When evaluating the model on a per-class basis, the highest precisions were attained for 1, 2, 4, 5, 6, and 8 leaves where the F1-score was 91%, 88%, 84%, 86%, 75%, and 77% respectively. Whereas for 3 and 7 (both having a low amount of training data), an accuracy of only 57% and 0% respectively were achieved.

Results indicate that the model estimates different development phases of weed plants with high precision; therefore, it is appropriate for usage in fields and will help in weed detection in blend with other support methods when performing weed control on farm fields.

References

[1] Telfer A, Bollman KM, Poethig RS. Phase change and the regulation of trichome distribution in Arabidopsis thaliana. Development. 1997;124(3):645-54.

[2] Minervini M, Scharr H, Tsaftaris SA. Image analysis: the new bottleneck in plant phenotyping [applications corner]. IEEE signal processing magazine. 2015;32(4):126-31.

[3] Spalding EP, Miller ND. Image analysis is driving a renaissance in growth measurement. Current opinion in plant biology. 2013;16(1):100-4.

[4] Aksoy EE, Abramov A, Wörgötter F, Scharr H, Fischbach A, Dellen B. Modeling leaf growth of rosette plants using infrared stereo image sequences. Computers and electronics in agriculture. 2015;110:78-90.

[5] Qawaqneh Z, Mallouh AA, Barkana BD. Age and gender classification from speech and face images by jointly fine-tuned deep neural networks. Expert Systems with Applications. 2017;85:76-86.

[6] Grinblat GL, Uzal LC, Larese MG, Granitto PM. Deep learning for plant identification using vein morphological patterns. Computers and Electronics in Agriculture. 2016;127:418-24.

[7] Dyrmann M, Karstoft H, Midtiby HS. Plant species classification using deep convolutional neural network. Biosystems engineering. 2016;151:72-80.

[8] Dyrmann M, Jørgensen RN, Midtiby HS. RoboWeedSupport-Detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. Advances in Animal Biosciences. 2017;8(2):842-7.

[9] dos Santos Ferreira A, Freitas DM, da Silva GG, Pistori H, Folhes MT. Weed detection in soybean crops using ConvNets. Computers and Electronics in Agriculture. 2017;143:314-24.

[10] Teimouri N, Dyrmann M, Nielsen PR, Mathiassen SK, Somerville GJ, Jørgensen RN. Weed growth stage estimator using deep convolutional neural networks. Sensors. 2018;18(5):1580.

[11] LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems. IEEE; 2010. p. 253-6.

[12] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning [http://www. deeplearningbook. org]. MIT Press, Cambridge, MA. 2016.

[13] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee; 2009. p. 248-55.

[14] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25.

[15] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 2818-26.

[16] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770-8.

[17] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014.

[18] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR; 2019. p. 6105-14.

[19] Ramachandran P, Zoph B, Le QV. Searching for activation functions. arXiv preprint arXiv:171005941. 2017.

[20] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014;15(1):1929-58.

[21] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.