



# Improving Deep Learning-based Plant Disease Classification with Attention Mechanism

Pendar Alirezazadeh<sup>1</sup> · Michael Schirrmann<sup>1</sup> · Frieder Stolzenburg<sup>2</sup>

Received: 10 May 2022 / Accepted: 14 November 2022 / Published online: 16 December 2022  
© The Author(s) 2022

## Abstract

In recent years, deep learning-based plant disease classification has been widely developed. However, it is challenging to collect sufficient annotated image data to effectively train deep learning models for plant disease recognition. The attention mechanism in deep learning assists the model to focus on the informative data segments and extract the discriminative features of inputs to enhance training performance. This paper investigates the Convolutional Block Attention Module (CBAM) to improve classification with CNNs, which is a lightweight attention module that can be plugged into any CNN architecture with negligible overhead. Specifically, CBAM is applied to the output feature map of CNNs to highlight important local regions and extract more discriminative features. Well-known CNN models (i.e. EfficientNetB0, MobileNetV2, ResNet50, InceptionV3, and VGG19) were applied to do transfer learning for plant disease classification and then fine-tuned by a publicly available plant disease dataset of foliar diseases in pear trees called DiaMOS Plant. Amongst others, this dataset contains 3006 images of leaves affected by different stress symptoms. Among the tested CNNs, EfficientNetB0 has shown the best performance. EfficientNetB0+CBAM has outperformed EfficientNetB0 and obtained 86.89% classification accuracy. Experimental results show the effectiveness of the attention mechanism to improve the recognition accuracy of pre-trained CNNs when there are few training data.

**Keywords** Plant disease classification · Deep learning · Attention mechanism · CBAM · Data limitation

---

✉ Pendar Alirezazadeh  
PAIirezazadeh@atb-potsdam.de

<sup>1</sup> Department of Engineering for Crop Production, Leibniz Institute for Agricultural Engineering and Bioeconomy (ATB), Max-Eyth-Allee 100, 14469 Potsdam, Germany

<sup>2</sup> Automation and Computer Sciences Department, Harz University of Applied Sciences, Friedrichstr. 57–59, 38855 Wernigerode, Germany

## Optimierung der auf Deep Learning basierenden Klassifizierung von Pflanzenkrankheiten mit CBAM

### Zusammenfassung

In den letzten Jahren werden immer stärker auf Deep Learning basierende Verfahren zur Klassifikation von Pflanzenkrankheiten entwickelt. Es bleibt jedoch herausfordernd, ausreichend annotierte Bilddatenbanken aufzubauen, um Deep-Learning-Modelle für die Erkennung von Pflanzenkrankheiten effektiv zu trainieren. Der Attention-Mechanismus im Deep Learning unterstützt das Modell dabei, sich auf die informativen Datensegmente zu fokussieren und die Unterscheidungsmerkmale der Eingabedaten zu extrahieren, um die Trainingsleistung zu verbessern. Diese Studie untersucht das Convolutional Block Attention Module (CBAM) zur Verbesserung der Klassifizierung mit CNNs, einem zusätzlichen Modul, das mit vernachlässigbarem Overhead in jede CNN-Architektur integriert werden kann. CBAM wird innerhalb von CNNs verwendet, um wichtige lokale Regionen hervorzuheben und diskriminierende Merkmale zu extrahieren. In dieser Studie wurden bekannte CNN-Modelle (d.h. EfficientNetB0, MobileNetV2, ResNet50, InceptionV3 und VGG19) genutzt, um Transfer-Lernen für die Klassifizierung von Pflanzenkrankheiten durchzuführen, und dann durch einen öffentlich verfügbaren Pflanzenkrankheitsdatensatz von Blattkrankheiten bei Birnenblättern (DiaMOS Plant) modelliert. Dieser Datensatz enthält unter anderem 3006 Bilder von Birnenblättern, die verschiedene Stresssymptomen aufweisen. Unter den getesteten CNNs zeigte EfficientNetB0 die beste Leistung. Durch die Integration von CBAM (EfficientNetB0+CBAM) konnte diese Leistung übertroffen und eine Klassifizierungsgenauigkeit von 86,89% erreicht werden. Experimentelle Ergebnisse zeigen, dass die Nutzung von CBAM zu einer Verbesserung der Erkennungsgenauigkeit von vortrainierten CNNs führen, insbesondere im Fall wenn nur wenige Trainingsdaten vorhanden sind.

**Schlüsselwörter** Klassifizierung von Pflanzenkrankheiten · Deep Transfer Learning · Attention-Mechanismus · CBAM · Datenlimitierung

### Introduction

Plant leaves contain valuable information about plant health. Because the first symptoms of plant stresses show up in leaves, the visual assessment of plant leaves is a valuable aid in the early detection of plant diseases and the prevention of crop failure. Yet, leaf disease identification by agricultural experts based on their experience is a time-consuming, tedious and inefficient process. On the other hand, blind use of pesticides in some cases not only does not prevent diseases but can also affect the quality of the product and lead to environmental pollution.

The advancement of deep learning techniques, particularly convolutional neural networks (CNNs) has become increasingly popular in precision agriculture Schirrmann et al. (2021); de Camargo et al. (2021). In recent years, automatic classification of crop diseases has been carried out using deep learning models to compensate for the lack of human expertise. Selvaraj et al. (2019) trained three different CNNs (ResNet50, InceptionV2, and MobileNetV1) using transfer learning to classify 18 different diseases using 18000 images of different parts of the banana plant. It was found that ResNet50 and InceptionV2 based models performed better compared to MobileNetV1 in their dataset. Kumar et al. (2020) explored InceptionV3 model to diagnose coffee leaf diseases. They collected a dataset of 1747 images of coffee leaves and categorized them into 5 classes (healthy and four different diseases). They utilized the transfer learning technique to reduce the train-

ing time. To overcome the problem of limited data, they applied data augmentation techniques to increase the data set used to train the network. Liu et al. (2020) generated a dataset of 107366 grape leaf images using image enhancement and data augmentation techniques. They proposed a Dense Inception-based Convolutional Neural Network (DICNN) to classify images into seven different classes. Ramcharan et al. (2017) collected a dataset of cassava disease images including 11670 images of three diseases and two types of pest damages. They applied transfer learning to train InceptionV3 model and analyzed the performance of the model with three different classifiers: the original softmax, support vector machines (SVM), and k-nearest neighbor (KNN). Liu and Wang (2020) created a dataset of tomato diseases and pests under real natural conditions with 15000 images for 12 different classes of diseases and pests. They optimized the feature layer of You Only Look Once version 3 (YoloV3) model using an image pyramid to achieve multi-scale feature detection and improve detection accuracy. Fuentes et al. (2017) considered three detectors: Faster Region-based Convolutional Neural Network (Faster R-CNN), Region-based Fully Convolutional Network (R-FCN), and Single Shot Multibox Detector (SSD) to explore the performance of Deep Learning in detecting diseases and pests on tomato. To improve the detection and localization of bounding boxes, they combined each of the detection models with deep feature extractors such as VGG network and Residual Network (ResNet). They evaluated their proposed method on a dataset of tomato diseases and pests,

which includes 5000 images of nine different classes of diseases and pests. Liu et al. (2017) proposed an AlexNet-based deep model for apple leaf disease detection using a dataset of 13689 images of four different apple disease. In addition to working on specific species, some researchers also focused on classifying different diseases in different species. Ferentinos (2018) explored different CNN architectures such as AlexNet, VGG and GoogleNet to classify 58 different plant diseases from 25 different plant species. Too et al. (2019) conducted an experiment to evaluate VGG16, InceptionV4, ResNet50-152 layers, and DenseNet121 for classifying 38 different diseases, including diseased and healthy images of leaves from 14 plants from the plantVillage dataset.

With the advances that have been made through the use of Deep Learning, some new challenges have also emerged. Deep Learning techniques require large amounts of data to train the network, which is virtually impossible to obtain due to the limited amount of annotated data in plant disease classification problems. In most cases, researchers fine-tune off-the-shelf deep models to solve the problem of limited data. Contrary to popular belief, some research has shown that transfer learning does not always lead to better performance because features cannot be readily transferred to other tasks Raghu et al. (2019); He et al. (2019). In addition, some methods try to improve the performance of the deep model by using complex scenarios and huge deep learning models. However, if we have a limited data problem, using a huge deep model with many parameters can lead to an overfitting problem. On the other hand, most models learn not only relevant disease features, but unfortunately they also learn irrelevant image features such as background noise or uninfected plant parts Ferentinos (2018); Mohanty et al. (2016); Toda and Okura (2019); Lee et al. (2020b). This will lead to confusion between similar plants of different disease classes. Therefore, the problem of limited data leads the model to learn irrelevant features. To solve this problem, Fuentes et al. (2017, 2019) proposed a region-based deep neural network to focus on contaminated parts of leaves. This is a very time-consuming technique because it requires labour-intensive manual annotation of disease locations and also depends heavily on prior knowledge of plant diseases. Lee et al. (2020a) developed a new method based on GoogleNet and Recurrent Neural Network (RNN) to automatically locate infected regions and extract relevant features for disease image classification (20 diseases and one healthy class). However, they performed better with GoogleNet compared to a combination of GoogleNet with RNN in the PlantVillage dataset. Using oversized deep neural network models tends to produce a lot of redundant features that are either shifted versions of one another or are very similar and reduce system performance Ayinde et al. (2019).

Because the size/shape of a leaf disease may be significantly different at different growth stages, the attention mechanism can enhance disease feature extraction by highlighting disease information while suppressing non-disease features of leaves and understating background information, resulting in better detection accuracy. The Convolutional Block Attention Module (CBAM) is an effective attention module for feedforward convolutional neural networks Woo et al. (2018). Given an intermediate feature map, CBAM sequentially infers the attention map along two separate dimensions (channel and spatial) and then multiplies the attention map with the input feature map for adaptive feature refinement. CBAM is a lightweight module with negligible parameters that can be plugged into the output of any CNN. In terms of both parameters and computations, the overall overhead of CBAM is quite small, but its role in improving the performance of CNNs is remarkable.

In this research, some well-known CNN architectures such as EfficientNetB0, MobileNetV2, ResNet50, InceptionV3, and VGG19 were trained with images of leaves of healthy and diseased plants to develop an automatic system for plant disease diagnosis. To solve the problem of limited data and learning irrelevant features, we plugged CBAM into the output feature map of CNNs to highlight important local regions and extract discriminative features. To demonstrate the effectiveness of the proposed system in a real-world application, we trained and tested our system using the DiaMOS Plant dataset, which contains images acquired under real field growing conditions Fenu and Mallocci (2021). The DiaMOS Plant dataset is an available dataset of leaf diseases in pear trees. Amongst others, it contains 3006 images of leaves showing different stress symptoms. The main contributions of the proposed work can be summarized as follows:

- We show that using a huge deep learning model with many parameters in small datasets is not an efficient solution.
- We explore the advantage of the attention module for learning representations of plant diseases in classification performance.
- We show the effectiveness of the attention mechanism to extract more discriminative features and improve the performance of pre-trained CNNs with little training data.

The rest of the paper is organized as follows. Sect. 2 introduces the CNN models, the CBAM that was used for the investigation of the attention mechanism Vaswani et al. (2017) that affects the performance of the system and proposed system. Sect. 3 presents the dataset used for training and testing, as well as the training procedure and performance evaluation. Sect. 4 presents the results of applying the proposed models for plant disease detection and diag-

**Table 1** A brief comparison between the CNN architectures used in this study

| Network        | Number of parameters | Number of layers | Input images' size        |
|----------------|----------------------|------------------|---------------------------|
| EfficientNetB0 | 5.3M                 | 18               | $224 \times 224 \times 3$ |
| MobileNetV2    | 3.5M                 | 53               | $224 \times 224 \times 3$ |
| InceptionV3    | 24M                  | 48               | $224 \times 224 \times 3$ |
| ResNet50       | 25M                  | 50               | $224 \times 224 \times 3$ |
| VGG19          | 143M                 | 19               | $224 \times 224 \times 3$ |

nosis, while the paper concludes in Sect. 5 with conclusions and future research to improve the proposed method.

## Methodology

### Pre-trained CNNs

In this study, we used the following five pre-trained CNNs for plant disease classification. We have initialized the networks with the weights from ImageNet Russakovsky et al. (2015) dataset and then frozen all the convolutional and max-pooling layers so that their weights could not be modified. Table 1 presents a brief information about the architecture of each network.

#### EfficientNet

Using the concept of compound scaling, Tan and Le (2019) proposed EfficientNet to create various models in the family that has great capability of feature extraction. EfficientNet is designed based on multiobjective neural architecture search, and composite scaling to regularly measure the depth, width, and resolution of the network. The core component of the network is a mobile inverted bottleneck convolution module which is inspired by inverted residual and residual structure. In EfficientNet, instead of scaling only the depth of the network, the network's width and resolution are scaled uniformly. Compared to other CNNs architectures, it has fewer parameters and higher accuracy. In this study, we used EfficientNetB0 which has 2 convolution layers and 16 mobile inverted bottleneck convolution modules. The total number of parameters for the whole network is about 5.3 million. EfficientNetB0 has a predefined  $224 \times 224 \times 3$  input size.

#### MobileNet

By replacing the standard convolution layers with depthwise separable convolution blocks, Howard et al. (2017) proposed MobileNet as an efficient and lightweight deep neural network to be used in mobile applications. Unlike standard convolution layers which include  $3 \times 3$  convolution layer followed by batch normalization and ReLU, in MobileNet each convolution layer split into a  $3 \times 3$  depthwise

convolution layer and a  $1 \times 1$  pointwise convolution layer. Depthwise convolution layer filters the inputs and pointwise convolution layer combines the filtered values into a new set of outputs. The first version of MobileNet (MobileNetV1) has a convolution layer and 13 depthwise separable convolution blocks. Based on the bottleneck depth-separable convolution with residuals, Sandler et al. (2018) proposed the second version of MobileNet called MobileNetV2. The main advantage of MobileNetV2 is each block has an extra  $1 \times 1$  expansion layer to expand the data (increase the number of channels). The residual connection in the bottleneck residual block works the same as in ResNet. We used MobileNetV2 with 3.4 M parameters for our experiments that includes the initial fully convolution layer with 32 filters and 19 residual bottleneck layers.

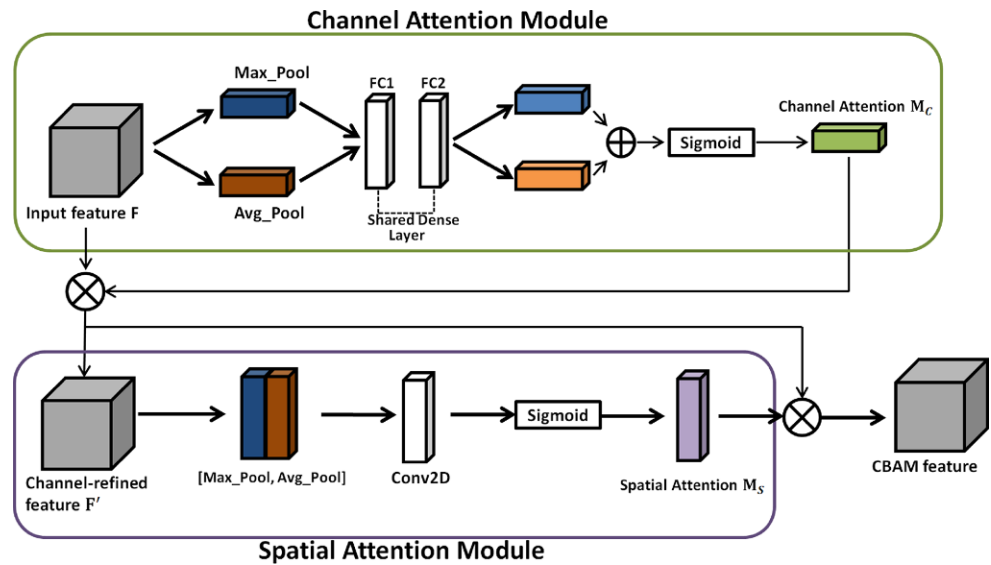
#### Inception

Szegedy et al. (2017) proposed a new type of architecture called GoogleNet or Inception. Inception networks try to improve computational efficiency while scaling up the network. The network heavily utilizes NiN (network in network) in its internal architecture to reduce the input dimension and to increase the network depth along with its width to improve the overall performance. The filter-level sparsity blocks are introduced in the inception module with the filter sizes of  $1 \times 1$ ,  $3 \times 3$ , and  $5 \times 5$ . The outputs of all layers are concatenated into one output vector. In this work, we used the third version of Inception (InceptionV3) which contains 24 million parameters and 48 layers.

#### ResNet

To avoid network complexity by increasing network depth and thus reducing system performance, the Residual Network architecture (ResNet) has been proposed He et al. (2016). ResNet is composed of several residual blocks with shortcut connections between layers. ResNet has been developed with a number of different layers (i.e. 50, 101, etc.). We chose ResNet-50 for our experiments which contains 49 convolution layers and 1 fully connected layer. The total number of parameters for the whole network is about 25 million. ResNet-50 has a predefined  $224 \times 224 \times 3$  input size.

**Fig. 1** The Convolutional Block Attention Module (CBAM). The upper side is the channel attention module, and the lower side is the spatial attention module



## VGG

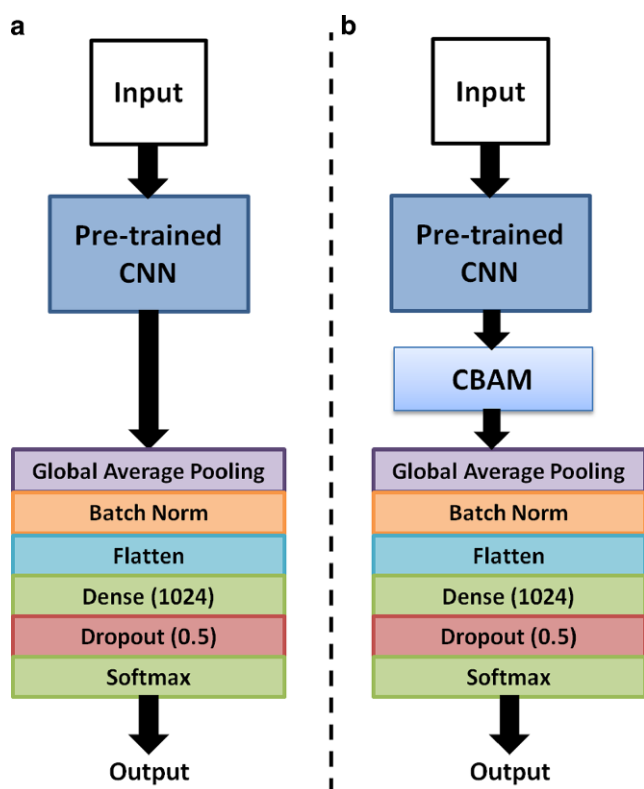
Simonyan and Zisserman (2014) investigated the effect of increased convolutional network depth on the performance of deep model and proposed VGGNET. VGGNET is composed of some convolution layers and fully connected layers. To reduce the number of parameters in VGGNET, convolution layers are implemented with small  $3 \times 3$  filters. Based on the number of convolution layers, two versions of VGGNET called VGG16 and VGG19 were released which have 13 and 16 convolution layers, respectively. Compared to other pre-trained models, VGGNET has a high number of parameters which makes it computationally expensive. In this study, we used VGG19 which contains 143M parameters.

## Convolutional block attention module (CBAM)

Oversized deep neural network models tend to produce a lot of redundant and irrelevant features on small datasets that are either shifted versions of one another or are very similar. Using a computationally efficient architecture, Woo et al. (2018) proposed CBAM to exploit both spatial and channel-wise attention Chen et al. (2017). The spatial attention is a process to pay attention by emphasizing the locations of percepts with high saliency to select relevant information. The channel-wise attention is a process that tries to apply weights to the channels based on their importance to highlight more important channels. CBAM contains two sequential submodules called the Channel Attention Module and the Spatial Attention Module and can be used for feed-forward CNN models. These two modules are applied in a particular order to examine the feature map extracted by the lower layers. The architecture of both modules are

shown in Fig. 1. The incoming feature map to the CBAM is transferred into the feature map  $F \in R^{C \times H \times W}$  where  $H$  and  $W$  denote the height and width of the feature map and  $C$  denotes the number of channels. In the following, the spatial dimension from the input map is removed by maximum and average pooling layers. The global average pooling layer obtains the aggregated spatial information, whereas the global max pooling layer captures distinctive object features to infer finer channel-wise attention. Using two shared dense layers, CBAM computes a channel attention map  $M_C \in R^{C \times 1 \times 1}$  from the reduced map (channel attention module, upper side of Fig. 1). Then, a channel refined feature map is obtained by multiplying the channel attention map by the incoming feature map  $F$ , such that each element in  $F$  is multiplied by the corresponding channel weight in the channel attention map. Compared to the channel attention module, the spatial attention module laid more emphasis on the parts of the feature map, which is complementary to the channel attention. To compute the spatial attention module  $M_S \in R^{1 \times H \times W}$  (lower side of Fig. 1), the channel refined feature map is squeezed into two 2D feature maps using maximum and average pooling along the channel axis. The two 2D feature maps are then concatenated and convolved by a standard convolution layer. To yield the final output of the CBAM, each element in the channel-refined map is multiplied with the weight by the corresponding spatial weight in the spatial attention map. As a result, CBAM can focus on ‘what’ and ‘where’ to highlight using the channel and the spatial attention operations, respectively. Thus, CBAM learns the key information in channel dimension and the spatial dimension to extract more efficient features.





**Fig. 2** The overall architecture of the models used in this study. The left side (a) is the pre-trained model and the right side (b) is the pre-trained model integrated with CBAM

### Proposed Model

The pre-trained CNNs have been implemented for the ImageNet classification problem, which includes 1000 categories of objects. Therefore, we transferred all their layers except the fully connected (FC) layers and replaced them with the FC layers including the global average pooling layer, batch normalization, flatten, dense, dropout, softmax layer, and a classification output layer which corresponds to the classes of our dataset (i.e. 4 classes of plant diseases). To show the effectiveness of CBAM, that it extracts more discriminative features and improves the performance of the pre-trained CNNs, we plugged the CBAM to the output feature map of the CNNs. Then, we applied the FC layers in the same way as the pre-trained models. Fig. 2 provides overviews of the pre-trained model and the pre-trained model integrated with CBAM (pre-trained+CBAM) architectures. We applied CBAM to the output feature map of the pre-trained model to extract more efficient features.

**Table 2** Number of images which were available in each class after labeling the 3006 leaf images

| Leaf Symptoms | Number of Images |
|---------------|------------------|
| Healthy       | 43               |
| Spot          | 884              |
| Curl          | 54               |
| Slug          | 2025             |

### Experimental setup

#### Image dataset

In this study, we used a field dataset to diagnose and classify plant symptoms called DiaMOS Plant Fenu and Mallocci (2021). DiaMOS Plant is a pilot dataset containing images of an entire growing season of pear trees, which includes 3,006 leaf images. Using different devices including a smartphone (Honor 6x) and DSRL camera (Canon EOS 60D), the images were collected from three pear trees of the same plot in Italy (2021) with two types of resolutions, at  $2976 \times 3968$  and  $3456 \times 5184$ , respectively. The leaf images were captured from the upper side of the leaf in uncontrolled conditions including various lighting, angle, background, and noises. These characteristics lead to a dataset that can be used for a real application. An expert has labeled biotic and abiotic stresses of each original image of the entire leaf into one of the four categories: leaf spot (caused by bacteria and fungi), leaf curl (multiple causes e.g. pests, diseases), slug damage (*Caliroa cerasi*), and healthy leaf. In Table 2 the number of images is given which were available in each class after labeling the 3006 leaf images.

#### Training procedure and performance evaluation

In order to have an unbiased comparison, we adopted the same strategy for training and evaluation of the pre-trained and the pre-trained+CBAM models. The experimental framework was written in Python 3.7 and the Keras deep learning 2.4.3 library based on TensorFlow 2.2.1 backend. Models were executed on a single NVIDIA GeForce RTX 2080Ti GPU with 11 GB of video memory. In Fenu and Mallocci (2021), the DiaMOS Plant dataset was divided into training, validation, and test sets with a ratio of 7 : 2 : 1, respectively. To preserve the percentage of samples for each class, the authors of the DiaMOS Plant split the dataset into training, validation, and test sets with the ShuffleSplit strategy provided by the scikit-learn 0.23.2 library. The ShuffleSplit strategy does not guarantee that all folds will be different Pedregosa et al. (2011), so it is probable that there are overlaps between training and testing sets. After a thorough investigation, we found that our hypothesis is

correct and there are about 200 overlaps between training and test sets. To prevent overlapping and invalid comparison between models, we performed ten-fold cross-validation. The complete dataset was split into ten folds. Then, we iteratively trained the deep learning models on nine folds while using the remaining holdout fold as the test set. The fitting procedure performed a total of ten times. To train the models, we set the number of epochs and the batch size to 100 and 32, respectively. As the dataset includes a limited number of images and to prevent overfitting and manage the unbalance of the classes, we applied data augmentation, including horizontal and vertical mirroring, rotation, color variation, and all images were resized to  $224 \times 224 \times 3$ . The input images of the CNNs underwent one-to-one augmentation without duplication. To avoid a long training time, we used pre-trained models trained using the ImageNet dataset with a cross-entropy function. To avoid the plateau phenomenon, the model’s validation loss was monitored during the reduction of the learning rate to stop it when it does not improve. A learning-rate of  $2 \times 10^{-5}$  and a momentum of 0.9 were set. We used the Adam algorithm for optimization. The performance of the models was evaluated by macro-averaged and micro-averaged versions of precision, recall, F-score, and overall classification accuracy extracted by the confusion matrices as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations, whereas Recall (Sensitivity) is the ratio of correctly predicted positive observations to all observations in the actual class. Thus, precision focuses on the prediction, whereas recall focuses on the measurements. F-Score is the harmonic average of Precision and Recall. The measure selected by the authors for ranking the systems was the overall classification accuracy score.

### Results and Discussion

The results obtained by the five pre-trained CNNs for classification of the pear disease images with and without applying CBAM to the output of CNNs are presented in the Table 3. The values are the means over the ten folds. We

**Table 3** Micro and macro measurements of the pre-trained and the pre-trained+CBAM models on the test set for classification of the “healthy”, “slug”, “curl” and “spot” classes in ten fold cross validation (mean ± standard deviation). The best results are indicated in bold for each network

| Network             | Macro (%)             |                      |                      |                      | Micro (%)            |                      |                      |                      | Acc |
|---------------------|-----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|-----|
|                     | Pr                    | Re                   | F1                   | F1                   | Pr                   | Re                   | F1                   | F1                   |     |
| ResNet50            | 47.00 ± 0.127         | 38.53 ± 0.116        | 40.54 ± 0.121        | 68.53 ± 0.183        | 69.76 ± 0.178        | 67.17 ± 0.223        | 69.90 ± 0.182        | 69.90 ± 0.182        |     |
| ResNet50+CBAM       | <b>48.24 ± 0.230</b>  | <b>39.06 ± 0.060</b> | <b>41.08 ± 0.073</b> | <b>70.50 ± 0.308</b> | <b>71.35 ± 0.169</b> | <b>68.33 ± 0.124</b> | <b>71.36 ± 0.149</b> | <b>71.36 ± 0.149</b> |     |
| VGG19               | 64.45 ± 3.261         | 64.85 ± 4.456        | 64.43 ± 3.807        | <b>72.63 ± 0.180</b> | 73.02 ± 0.565        | <b>72.64 ± 0.156</b> | 73.09 ± 0.235        | 73.09 ± 0.235        |     |
| VGG19+CBAM          | <b>68.1 ± 6.953</b>   | <b>64.76 ± 4.694</b> | <b>65.97 ± 5.626</b> | 72.50 ± 0.559        | <b>73.36 ± 0.568</b> | 72.59 ± 0.484        | <b>73.42 ± 0.332</b> | <b>73.42 ± 0.332</b> |     |
| InceptionV3         | 62.60 ± 0.031         | 61.71 ± 0.157        | 61.96 ± 0.127        | 75.67 ± 0.160        | 74.44 ± 0.186        | 74.80 ± 0.234        | 76.61 ± 0.182        | 76.61 ± 0.182        |     |
| InceptionV3+CBAM    | <b>78.13 ± 13.872</b> | <b>65.45 ± 4.734</b> | <b>66.73 ± 7.035</b> | <b>77.38 ± 0.944</b> | <b>77.24 ± 0.356</b> | <b>77.30 ± 0.631</b> | <b>77.87 ± 0.446</b> | <b>77.87 ± 0.446</b> |     |
| MobileNetV2         | 66.47 ± 0.141         | 66.54 ± 0.185        | 66.49 ± 0.171        | 81.75 ± 0.273        | 81.96 ± 0.249        | 81.79 ± 0.255        | 82.06 ± 0.235        | 82.06 ± 0.235        |     |
| MobileNetV2+CBAM    | <b>70.68 ± 0.453</b>  | <b>70.00 ± 0.572</b> | <b>70.27 ± 0.544</b> | <b>83.36 ± 0.125</b> | <b>83.58 ± 0.136</b> | <b>83.66 ± 0.131</b> | <b>83.99 ± 0.149</b> | <b>83.99 ± 0.149</b> |     |
| EfficientNetB0      | 67.38 ± 0.256         | 67.82 ± 0.121        | 67.59 ± 0.191        | 85.44 ± 0.289        | 85.41 ± 0.384        | 85.66 ± 0.344        | 85.82 ± 0.379        | 85.82 ± 0.379        |     |
| EfficientNetB0+CBAM | <b>67.88 ± 0.263</b>  | <b>68.09 ± 0.099</b> | <b>67.98 ± 0.171</b> | <b>86.12 ± 0.275</b> | <b>86.31 ± 0.327</b> | <b>86.41 ± 0.297</b> | <b>86.89 ± 0.332</b> | <b>86.89 ± 0.332</b> |     |

**Table 4** Comparison of different networks in terms of parameters and computation over ten fold cross validation (mean  $\pm$  standard deviation) when we integrated them to CBAM. CBAM has a light overhead and computational load

| Network             | Number of parameters | Training time (s/epoch) | Classification time (ms/image) |
|---------------------|----------------------|-------------------------|--------------------------------|
| ResNet50            | 25 698 180           | 116.3 $\pm$ 2.46        | 32.3 $\pm$ 1.84                |
| ResNet50+CBAM       | 26 749 158           | 123.8 $\pm$ 1.96        | 32.7 $\pm$ 1.47                |
| VGG19               | 143 555 844          | 165.5 $\pm$ 2.76        | 46.7 $\pm$ 2.24                |
| VGG19+CBAM          | 144 622 054          | 173.3 $\pm$ 3.13        | 48.1 $\pm$ 2.94                |
| InceptionV3         | 23 913 252           | 118.6 $\pm$ 3.24        | 33.5 $\pm$ 2.23                |
| InceptionV3+CBAM    | 24 964 230           | 121.4 $\pm$ 2.68        | 34.8 $\pm$ 1.91                |
| MobileNetV2         | 3 578 948            | 96.7 $\pm$ 3.65         | 24.5 $\pm$ 2.34                |
| MobileNetV2+CBAM    | 3 990 086            | 99.4 $\pm$ 1.43         | 26.5 $\pm$ 1.97                |
| EfficientNetB0      | 5 370 528            | 86.6 $\pm$ 1.64         | 22.1 $\pm$ 1.33                |
| EfficientNetB0+CBAM | 5 781 666            | 92.3 $\pm$ 2.12         | 22.9 $\pm$ 1.55                |

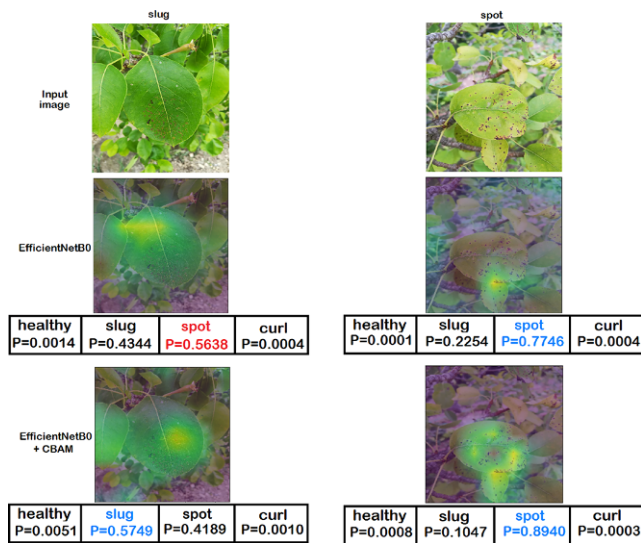
compared the performance of applying CBAM on the output of ImageNet pre-trained models including ResNet50, VGG19, InceptionV3, MobileNetV2 and EfficientNetB0 since they were reported to be the baseline models for pear disease identification on the DiaMOS Plant dataset Fenu and Mallocci (2021). As we have discussed in Sect. 3.2, there are overlaps between training and test sets provided by DiaMOS Plant dataset collectors. This is the main reason why they have achieved higher rates than us. Therefore, for an unbiased comparison, we trained and tested the models on disjoint sets over ten-fold cross validation. First, we compared the performances of the three most depth models i.e. ResNet50, VGG19 and InceptionV3. The ResNet50, VGG19 and InceptionV3 achieved accuracies of 69.90, 73.09 and 76.61%, respectively. The ResNet50, VGG19 and InceptionV3 are very huge and heavy models that have many parameters to learn. On the other hand, DiaMOS Plant dataset is an imbalanced small dataset with 3,505 images, which includes 43 and 54 images for healthy and curl classes, respectively. As we discussed earlier, using a huge deep learning model with many parameters on a small dataset can lead to an overfitting problem and consequently reduce system performance in the test set. To demonstrate the role of CBAM in representation power enhancement and performance improvement, CBAM was only plugged into the baseline networks. The networks integrated with CBAM outperform the baselines in terms of all the performance metrics except for VGG19. VGG19+CBAM has close performance with VGG19 for micro-averaged precision and micro-averaged F-score metrics. This could be due to the large size of VGG19 for DiaMOS Plant dataset. However, the networks with CBAM achieved the best accuracies compared to the baselines, demonstrating that the CBAM generates a richer descriptor and spatial attention that complements the channel attention effectively. Applying CBAM to the output of the models improved the accuracies of ResNet50, VGG19, and InceptionV3 by 1.46, 0.33, and 1.26%, respectively. Since the overall overhead of CBAM is quite small in terms of both parameters and

computation as shown in the Table 4, CBAM was also applied to the light-weight networks, MobileNetV2 and EfficientNetB0. MobileNetV2 and EfficientNetB0 achieved accuracies of 82.06 and 85.82%, respectively, indicating that using the light-weight backbone networks for the small DiaMOS Plant dataset leads to better performances. As shown in the Table 3, we observe significant improvements from MobileNetV2 and EfficientNetB0 for all performance metrics, demonstrating the effect of applying CBAM to the baseline methods. MobileNetV2+CBAM and EfficientNetB0+CBAM achieved accuracies of 83.99 and 86.89%, respectively, which are greater than MobileNetV2 and EfficientNetB0 by 1.93 and 1.07%, respectively. Compared to the baselines, EfficientNetB0 has the best results for all performance metrics. EfficientNetB0 integrated with CBAM outperforms EfficientNetB0 and improves macro-averaged and micro-averaged versions of precision, recall, F-score, and overall classification accuracy.

To make the role of CBAM in discriminative feature enhancement and performance improvement more transparent, gradient-weighted class activation maps (Grad-CAM) Selvaraju et al. (2017) were applied to EfficientNetB0 and EfficientNetB0+CBAM networks using images from the DiaMOS Plant test set to highlight important regions. Grad-CAM is a gradients-based visualization method that tries to calculate the importance of the spatial locations in convolutional layers with respect to a unique class. We attempted to look at how CBAM helps the network to enhance the power of discrimination by highlighting the regions that the network has considered as important for predicting a class. The visualization results of CBAM-integrated EfficientNetB0 (EfficientNetB0+CBAM) were compared with baseline (EfficientNetB0). Fig. 3 illustrates the visualization results. The Softmax scores for target class and also different classes are shown in the figure.

From Fig. 3, it can be seen that EfficientNetB0+CBAM covers the plant symptoms regions better than EfficientNetB0. CBAM helps EfficientNetB0 to better exploit information in leaf diseases regions and aggregate features from





**Fig. 3** Grad-CAM visualization results that highlight the importance regions for trained model prediction. We compared the visualization results of CBAM-integrated EfficientNetB0 (EfficientNetB0+CBAM) with baseline (EfficientNetB0). The grad-CAM visualization was calculated for the last convolutional outputs. The ground-truth label is shown on the top of each input image and P denotes the softmax score of each network for the different classes. The correctly predicted class and its score are shown in blue and the incorrectly predicted class and its score are shown in red. It is apparent that CBAM supports the network to correct its prediction and increase target class scores

them. Due to the high degree of similarity between leaf spot and slug damage in some images, the network predicts the target class incorrectly. As shown in Fig. 3, EfficientNetB0 predicted the slug input image as leaf spot class. The feature refinement process of CBAM helps the network to utilize given features well and correct its prediction. In addition, for the leaf spot input, CBAM helps the network to increase the target class score and decrease other class scores accordingly. This leads to a more discriminative deep

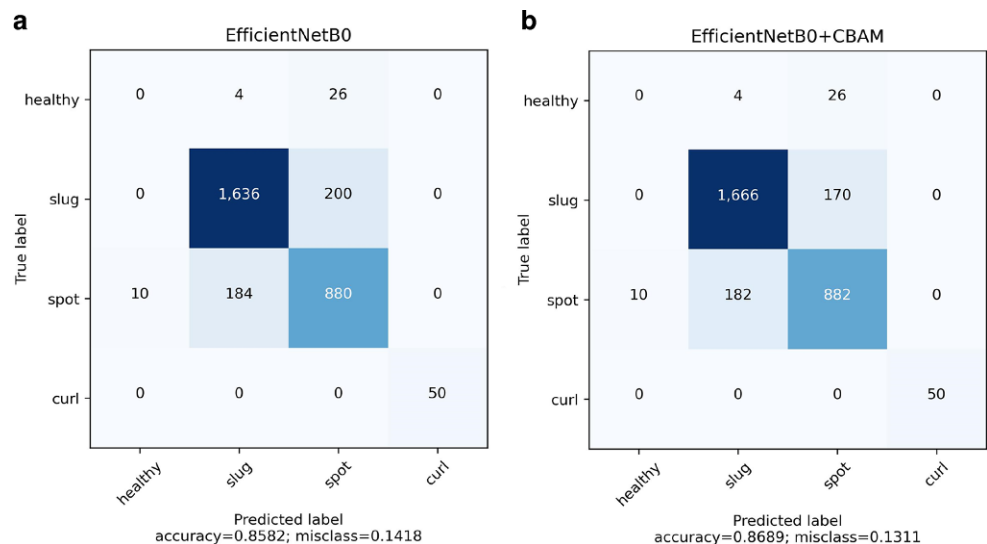
learning model, which can help classifying real application data.

Confusion matrices related to the performance of EfficientNetB0 (as the best performed pre-trained CNN) and EfficientNetB0+CBAM are shown in Fig. 4. The confusion matrices obtained by summing confusion matrices of all the ten folds. The confusion matrix provides the performance of a predictive model to show which classes are being predicted correctly and which are incorrectly predicted. As we can see, EfficientNetB0 predicted some slug damage images as leaf spot class because of the high degree of similarity and imbalanced data. Integrating the network with CBAM helps to distinguish between classes and improve classification accuracy.

It is interesting that these improvements result from plugging CBAM into the pre-trained models with a negligible parameter overhead, indicating that enhancement is not due to a naive capacity increment but because of CBAM’s effective feature refinement. As a result, the networks integrated with CBAM outperform all the baselines, demonstrating the general applicability of CBAM across different architectures. CBAM can be seamlessly integrated and trained into any CNN architecture to improve networks. In addition, the result using the light-weight backbone networks (MobileNetV2 and EfficientNetB0) and the overhead of CBAM show that CBAM can be an effective module to improve the performance of networks in low-end devices.

To determine whether there were statistically significant differences between the mean of the pre-trained and the pre-trained+CBAM models in Table 3, the paired t-test was conducted. In the paired t-test, if  $p$ -Value  $< \alpha = 0.05$ , the null hypothesis is rejected, which means that differences in the model outcomes are so convincing at 95% of confidence that they can be considered significant. Table 5 reports the results of the paired t-test. According to Table 5,

**Fig. 4** Confusion matrices related to **a** EfficientNetB0 and **b** EfficientNetB0+CBAM for classifying plant disease images of the DiaMOS Plant dataset (obtained by summing the confusion matrices of all the ten folds)



**Table 5** The results of the paired t-test between pre-trained and proposed models

| Pre-trained model | Proposed model      | $T_{obs}$ | $p$ -Value |
|-------------------|---------------------|-----------|------------|
| ResNet50          | ResNet50+CBAM       | -6.5682   | 0.0003     |
| VGG19             | VGG19+CBAM          | -1.6273   | 0.1477     |
| InceptionV3       | InceptionV3+CBAM    | -2.4375   | 0.0449     |
| MobileNetV2       | MobileNetV2+CBAM    | -6.0106   | 0.0005     |
| EfficientNetB0    | EfficientNetB0+CBAM | -6.9361   | 0.0002     |

$p$ -Value < 0.05 were found in all comparisons except for VGG19. Using a huge deep learning model with many parameters such as VGG19 on a small dataset can lead to an overfitting problem and consequently reduce system performance in the test set. Therefore, CBAM plugging to VGG19 has no effect on improving the performance. From Table 5, the null hypothesis is rejected for ResNet50, InceptionV3, MobileNetV2, and EfficientNetB0, and it can be concluded that the resultant accuracy in Table 3 between the pre-trained and the pre-trained+CBAM models are not due to chance for the mentioned networks. According to Table 5, the hypothesis proposed in innovations is proven.

## Conclusions and Future Directions

In this study, the Convolutional Bottleneck Attention Module (CBAM) was investigated to improve the representation power of CNN networks for automatic plant disease classification. We applied attention-based feature refinement with two different modules, channel and spatial, to the baselines' output feature map to improve performance while keeping overhead low. CBAM helps baselines learn what and where to highlight to effectively refine features. We have shown that integrating CBAM with baselines has higher generalization ability than baselines, especially in discriminating similar symptom classes. We conducted our experiments on the DiaMOS Plant dataset, which was collected under uncontrolled conditions. The integrated baselines with CBAM performed better than all other baselines. We also showed that CBAM causes the network to properly focus on the target plant disease class. CBAM could help networks overcome the problem of lack of sufficient training data needed to learn deep models. In this study, we only plugged CBAM into the pre-trained models to show the important role of the attention module in improving plant disease classification in an imbalanced small dataset. Future research directions include: (1) developing a novel deep model based on the attention module, e.g., CBAM and residual or dense blocks with light weights such as SoilNet Alirezazadeh et al. (2021) for plant disease classification; (2) using a margin-based softmax loss Alirezazadeh et al. (2022); Tavakoli et al. (2021) instead of the original softmax to improve the discriminative power of the feature space.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Conflict of interest** P. Alirezazadeh, M. Schirrmann, and F. Stolzenburg declare that they have no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Alirezazadeh P, Rahimi-Ajdadi F, Abbaspour-Gilandeh Y et al (2021) Improved digital image-based assessment of soil aggregate size by applying convolutional neural networks. *Comput Electron Agric* 191:499. <https://doi.org/10.1016/j.compag.2021.106499>
- Alirezazadeh P, Dornaika F, Moujahid A (2022) Deep learning with discriminative margin loss for cross-domain consumer-to-shop clothes retrieval. *Sensors* 22(7):2660. <https://doi.org/10.3390/s22072660>
- Ayinde BO, Inanc T, Zurada JM (2019) Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks* 118:148–158. <https://doi.org/10.1016/j.neunet.2019.04.021>
- de Camargo T, Schirrmann M, Landwehr N et al (2021) Optimized deep learning model as a basis for fast uav mapping of weed species in winter wheat crops. *Remote Sens* 13(9):1704. <https://doi.org/10.3390/rs13091704>
- Chen L, Zhang H, Xiao J et al (2017) SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning, pp 6298–6306 <https://doi.org/10.1109/CVPR.2017.667>
- Fenu G, Mallocci FM (2021) Diamos plant: A dataset for diagnosis and monitoring plant disease. *Agronomy* 11(11):2107. <https://doi.org/10.3390/agronomy11112107>
- Ferentinos KP (2018) Deep learning models for plant disease detection and diagnosis. *Comput Electron Agric* 145:311–318. <https://doi.org/10.1016/j.compag.2018.01.009>
- Fuentes A, Yoon S, Kim SC et al (2017) A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors* 17(9):2022. <https://doi.org/10.3390/s17092022>
- Fuentes A, Yoon S, Park DS (2019) Deep learning-based phenotyping system with global description of plant anomalies and symptoms. *Front Plant Sci* 10:1321. <https://doi.org/10.3389/fpls.2019.01321>
- He K, Zhang X, Ren S et al (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*

- puter vision and pattern recognition, pp 770–778 <https://doi.org/10.1109/CVPR.2016.90>
- He K, Girshick R, Dollár P (2019) Rethinking imagenet pre-training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 4918–4927 <https://doi.org/10.1109/ICCV.2019.00502>
- Howard AG, Zhu M, Chen B et al (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications (arXiv preprint arXiv:170404861)
- Kumar M, Gupta P, Madhav P et al (2020) Disease detection in coffee plants using convolutional neural network. In: 2020 5th International Conference on Communication and Electronics Systems (ICES). IEEE, pp 755–760 <https://doi.org/10.1109/ICES48766.2020.9138000>
- Lee SH, Goeau H, Bonnet P et al (2020a) Attention-based recurrent neural network for plant disease classification. *Front Plant Sci* 11:1897. <https://doi.org/10.3389/fpls.2020.601250>
- Lee SH, Goeau H, Bonnet P et al (2020b) New perspectives on plant disease characterization based on deep learning. *Comput Electron Agric* 170:105220. <https://doi.org/10.1016/j.compag.2020.105220>
- Liu J, Wang X (2020) Tomato diseases and pests detection based on improved yolo v3 convolutional neural network. *Front Plant Sci* 11:898. <https://doi.org/10.3389/fpls.2020.00898>
- Liu B, Zhang Y, He D et al (2017) Identification of apple leaf diseases based on deep convolutional neural networks. *Symmetry* 10(1):11. <https://doi.org/10.3390/sym10010011>
- Liu B, Ding Z, Tian L et al (2020) Grape leaf disease identification using improved deep convolutional neural networks. *Front Plant Sci* 11:1082. <https://doi.org/10.3389/fpls.2020.01082>
- Mohanty SP, Hughes DP, Salathé M (2016) Using deep learning for image-based plant disease detection. *Front Plant Sci* 7:1419. <https://doi.org/10.3389/fpls.2016.01419>
- Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: Machine learning in python. *J Mach Learn Res* 12:2825–2830
- Raghu M, Zhang C, Kleinberg J et al (2019) Transfusion: Understanding transfer learning for medical imaging. In: Advances in neural information processing systems, p 32
- Ramcharan A, Baranowski K, McCloskey P et al (2017) Deep learning for image-based cassava disease detection. *Front Plant Sci* 8:1852. <https://doi.org/10.3389/fpls.2017.01852>
- Russakovsky O, Deng J, Su H et al (2015) Imagenet large scale visual recognition challenge. *Int J Comput Vis* 115(3):211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- Sandler M, Howard A, Zhu M et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520 <https://doi.org/10.1109/CVPR.2018.00474>
- Schirrmann M, Landwehr N, Giebel A et al (2021) Early detection of stripe rust in winter wheat using deep residual neural networks. *Front Plant Sci* 12:475. <https://doi.org/10.3389/fpls.2021.469689>
- Selvaraj MG, Vergara A, Ruiz H et al (2019) Ai-powered banana diseases and pest detection. *Plant Methods* 15(1):1–11. <https://doi.org/10.1186/s13007-019-0475-z>
- Selvaraju RR, Cogswell M, Das A et al (2017) Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision, pp 618–626 <https://doi.org/10.1109/ICCV.2017.74>
- Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition (arXiv preprint arXiv:14091556)
- Szegedy C, Ioffe S, Vanhoucke V et al (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence <https://doi.org/10.1609/aaai.v31i1.11231>
- Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning, PMLR, pp 6105–6114
- Tavakoli H, Alirezazadeh P, Hedayatipour A et al (2021) Leaf image-based classification of some common bean cultivars using discriminative convolutional neural networks. *Comput Electron Agric* 181:105935. <https://doi.org/10.1016/j.compag.2020.105935>
- Toda Y, Okura F (2019) How convolutional neural networks diagnose plant disease. *Plant Phenomics*. <https://doi.org/10.34133/2019/9237136>
- Too EC, Yujian L, Njuki S et al (2019) A comparative study of fine-tuning deep learning models for plant disease identification. *Comput Electron Agric* 161:272–279. <https://doi.org/10.1016/j.compag.2018.03.032>
- Vaswani A, Shazeer N et al (2017) Attention is all you need. In: Advances in neural information processing systems (<http://papers.nips.cc/paper/7181-attention-is-all-you-need>)
- Woo S, Park J, Lee JY et al (2018) Cbam: Convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV), pp 3–19 [https://doi.org/10.1007/978-3-030-01234-2\\_1](https://doi.org/10.1007/978-3-030-01234-2_1)

**Pendar Alirezazadeh** was born in 1987. He received his M.Sc. in Information Technology-Multimedia Systems from Razi University in 2015. He works as a research scientist at Leibniz Institute for Agricultural Engineering and Bioeconomy, Potsdam, Germany. His research focuses on the innovative deep learning techniques to improve deep learning-based agricultural image analysis.