JEE Journal of Ecological Engineering

Journal of Ecological Engineering 2023, 24(2), 61–71
https://doi.org/10.12911/22998993/156754
ISSN 2299–8993, License CC-BY 4.0

Received: 2022.10.22
Accepted: 2022.12.08
Published: 2023.01.01

Parallelization of Concise Convolutional Neural Networks for Plant Classification

Arnes Sembiring[1*], Yuwaldi Away[1,2], Fitri Arnia[1,2], Rusdha Muharar[1,2]

[1] Doctoral Program, School of Engineering, Universitas Syiah Kuala, Banda Aceh 23111, Indonesia
[2] Department of Electrical and Computer Engineering, Universitas Syiah Kuala, Banda Aceh 23111, Indonesia
* Corresponding author's email: arnessembiring4@gmail.com

ABSTRACT

Monitoring the agricultural field is the key to preventing the spread of disease and handling it quickly. The computer-based automatic monitoring system can meet the needs of large-scale and real-time monitoring. Plant classifiers that can work quickly in computer with limited resources are needed to realize this monitoring system. This study proposes convolutional neural network (CNN) architecture as a plant classifier based on leaf imagery. This architecture was built by parallelizing two concise CNN channels with different filter sizes using the addition operation. GoogleNet, SqueezeNet and MobileNetV2 were used to compare the performance of the proposed architecture. The classification performance of all these architectures was tested using the PlantVillage dataset which consists of 38 classes and 14 plant types. The experimental results indicated that the proposed architecture with a smaller number of parameters achieved nearly the same accuracy as the comparison architectures. In addition, the proposed architecture classified images 5.12 times faster than SqueezeNet, 8.23 times faster than GoogleNet, and 9.4 times faster than MobileNetV2. These findings suggest that when implemented in the agricultural field, the proposed architecture can be a reliable and faster plant classifier with fewer resources.

Keywords: parallelization of concise CNN, plant classification, multi-scale CNN.

INTRODUCTION

Monitoring the agricultural environment plays a major role in early detection of plant diseases so that further damage can be prevented and disease spread can be controlled more quickly and at a lower cost (Najdenovska et al., 2021). Monitoring, on the other hand, necessitates a large amount of manpower, takes a long time, and ultimately costs a lot (Ma et al., 2018). Computerized monitoring can significantly reduce these costs, while increasing monitoring efficiency and effectiveness (Lajoie-O'Malley et al., 2020).

The ability of a computer system to classify plant species and distinguish between healthy and disease-exposed plants is absolutely necessary for automated monitoring with a computer system (Knoll et al., 2018). Plant classifiers have been developed using a variety of algorithms, including Naive Bayes, random forest, support vector machine (SVM), K-nearest neighbor (KNN), decision trees, and artificial neural networks (ANNs). The role of leaf features as distinguishing description is important in the methods described above. With the advancement of computing technology, it is relatively simple to extract more than 100 features from a leaf, but it remains difficult to determine which features contribute the most to classification. Therefore, feature engineering (FE) and the presence of an expert are still required for this task (Zhang et al., 2019).

Since its introduction in 2012, the convolutional neural network (CNN) has outperformed other algorithms in classification in a variety of fields, with a classification accuracy close to 100% (Hassan et al., 2021; Mohanty et al., 2016). Furthermore, CNN can automatically extract important and unique features of each class from image and video data without the need for feature engineering or the presence of an expert

to select the most optimal features in the classification (Kamilaris and Prenafeta-Boldú, 2018; Yamashita et al., 2018). The advantages of CNN are driving its widespread use in the automotive, health, business, and other industries, including agriculture (Too et al., 2019).

However, implementing CNN as a classifier in agricultural monitoring systems faces numerous challenges, particularly in developing countries. These difficulties arise because the implementation of CNN necessitates large computing resources and a lengthy training period (Y. Wu et al., 2020), and farmers in developing countries are generally unable to provide adequate computing resources and internet connections to run CNN monitoring systems (Rahman et al., 2020). Therefore, trade-off between classification performance and computing resources is required so that CNN can be used in a low-cost monitoring system in agriculture (Karthik et al., 2020).

This study aimed to develop a concise CNN model by fusing two CNN channels with different filter sizes using an addition operation, as well as to provide a reliable CNN model that is faster than comparison architectures in classifying plants based on leaf images. This study also investigated the performance of the proposed CNN and comparison CNN models on plant classification using datasets with varying number of classes.

This work's main contributions are: (1) the proposed CNN model has fewer parameters and performs classification faster than all of the comparison architectures in this study. A smaller number of parameters will allow for less expensive implementation (2) the classification accuracy performance of the proposed CNN model is nearly identical to the classification accuracy performance of all comparison architectures in this study.

MATERIALS AND METHODS

Plant classification

Tomato diseases were identified using the SVM classifier in (Mokhtar et al., 2015). The dataset was classified into two classes, beginning with segmentation on each image. The classification accuracy of the SVM classifier with five different kernels was 92%. The SVM classifier was used again in (Kaur et al., 2018) to classify the soybean leaf images from the PlantVillage dataset with 90% accuracy. The SVM was built using a combination of texture and color features. Study in (Chouhan et al., 2018) classified six fungal diseases using a Radial Basis Function Neural Network (RBFNN). Bacterial foraging optimization (BFO) was used to improve the speed and accuracy of RBFNN, and this method outperforms the K-means (KM) and Genetic Algorithm (GA) algorithms.

The use of CNN in plant classification is dominated by the use of architectures that are known to be reliable, with AlexNet and VGG being the most widely used architectures (Abade et al., 2021). AlexNet and VGG architectures were used via transfer learning and fine tuning schemes in (Mohanty et al., 2016; Lu et al., 2017; Ferentinos, 2018; Suryawati et al., 2018; Rangarajan et al., 2018; Howlader et al., 2019; Luna et al., 2019). The classification accuracy of these two CNN models is generally around 99%. However, these two CNN models have a large number of parameters, with AlexNet having 62M and VGG having 138M. As a result, even if only for transfer learning and fine tuning schemes, these CNN models necessitate large computing resources, particularly when training from scratch (Rangarajan et al., 2018; Ferentinos, 2018).

CNN models with fewer parameters, such as googleNet with 6.8M, were used in (McCool et al., 2017) and (Maeda-Gutiérrez et al., 2020). Studies in (Elhassouny and Smarandache, 2019) and (Hassan et al., 2021) used Mobilenet-v2, which has a total of 3.4M parameters. Squeezenet with a parameter number of 1.24M was used for plant classification in (Aravind et al., 2020) and (Liu et al., 2021). The reliable performance of the three architectures in these studies confirms that a concise CNN architecture can reliably perform the classification task of datasets with relatively few classes.

Batch normalization

A normalized batch layer follows each convolution layer in the CNN architecture utilized in this work. Sergey Ioffe and Christian Szegedy introduced the use of batch normalization in CNN in 2015 to lessen internal covariate shift (ICS) during CNN training. CNN training toward convergence will be accelerated by reducing ICS (Ioffe and Szegedy, 2015).

Several studies cast doubt on the role of normalized batch layers in ICS reduction. The study in (Santurkar et al., 2018) shows that batch

normalization does not completely solve this ICS problem, although this study still confirms that batch normalization increases the speed of deep learning training to achieve convergence by controlling the mean and variance of the dataset.

The study in (Bjorck et al., 2018) also questions the contribution of ICS reduction to the success of batch normalization in expediting deep learning network training. The results of this study provide the evidence that batch normalization produces more reliable gradient updates, enabling deep learning networks to operate at greater learning rates and expediting the training of the network toward convergence.

The strengthening of Ioffe and Szegedy's argument is obtained from a study by (Awais et al., 2021). A series of experiments in this study show that ICS reduction is a major factor in increasing the convergence of a deep learning network, not only by batch normalization but also by all other methods that contribute to ICS reduction.

Batch normalization $\hat{x}_i$ can be determined by using Eq. 1:

$$\widehat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + e}} \tag{1}$$

where: μ_B adalah is the mean of the input x_i and σ_B^2 is the variance. The value of e is used to avoid division by zero when σ_B is very small so that numerical stability is increased. Furthermore, the final result of the y_i normalization batch is calculated by Eq. 2.

$$y_i = \gamma \widehat{x}_i + \beta \tag{2}$$

where: γ the scaling factor and β is the shifting factor and these two values are included in the parameters studied during CNN training.

Fusion of CNN channel

The various CNN architectures proposed recently use multiple modules consisting of convolution layer parallel channels. The use of channels with different filter sizes is intended to improve the ability of the CNN model to handle objects at multiple scales (Szegedy et al., 2015). Concatenation and addition operations can be used to combine two or more CNN channels. Concatenation is a channel-wise action that is more commonly employed than addition, which is an element-wise operation (Wu and Wang, 2019). While in the addition operation the size of the fused layer is the same as the initial size of the two channels, in concatenation the size of the fused layer is equal to the total of the sizes of the two channels. Equation 3 shows the concatenation operation on tensor A and tensor B which form tensor C, while Eq. 4 is the addition operation of tensor A and tensor B.

$$\mathrm{C} = [\mathrm{A} \quad \mathrm{B}] \tag{3}$$

$$\mathrm{C} = [\mathrm{A} + \mathrm{B}] \tag{4}$$

The CNN GoogleNet model uses 9 concatenation operations to combine multiple parallel channels on the inception module. The first channel contains 1 convolution layer with a filter size of 1 × 1, the second channel with a filter size of 3 × 3 and the third channel with a filter size of 5 × 5 is combined with the concatenation operation (Szegedy et al., 2015). Squeezenet uses 8 concatenation operations in its architecture (Iandola et al., 2016). MobileNetV2 uses 10 addition operations and ShuffleNet uses these two operations with 3 concatenation operations and 13 addition operations (Sandler et al., 2018). However, the addition operation on ShuffleNet and MobileNetV2 adds the feature maps from the previous convolution block to the feature map of the following block rather than the feature maps of the two parallel channels. In this work, the proposed approach is to combine the two feature maps that were created through the extraction of two parallel CNN channels with the same depth, but different filter sizes.

Dataset

The dataset used in this study contains leaf images that are grouped into 38 classes consisting of 14 classes from different healthy plants and 28 classes from leaves exposed to various diseases. All images in this dataset are from the open access repository via the PlantVillage project (Hughes and Salathe, 2015). The PlantVillage dataset is one of the most important datasets in the field of plant classification (Brahimi, 2018). A review of 121 plant classification studies (Abade et al., 2021) from 2010 – 2019 showed that 65% of the studies were conducted with stored datasets or in a controlled experimental environment. Out of these studies, 45 made use of the PlantVillage dataset. Some of the studies, like the one by (Mohanty et al., 2016) and (Saleem et al., 2020), tested the effectiveness of the CNN design using 38 already-existing classes. Others just employed

a few classes, such as (Brahimi et al., 2017) who employed nine classes of diseased tomato leaves and (Maeda-Gutiérrez et al., 2020) who employed ten classes of healthy and diseased tomato leaves.

Each image in the PlantVillage dataset was a single leaf RGB image with a size of 256 × 256. The total number of images in this dataset was 70.846 and was divided into 80% for training and 20% for testing. This dataset was utilized for both transfer learning on the comparison architectures and for training from scratch on the proposed CNN model. Image size was maintained at 256 × 256 in the proposed CNN model training. In transfer learning for the comparison architectures, the size of this image was changed according to the default size of the input image of each comparison architecture. In both training and testing schemes, there was no further image preprocessing applied to the dataset.

The proposed architecture

Figure 1 shows the proposed architecture. The CNN model called SlimPlantNet was built from the fusion of two CNN channels with different filter sizes in some convolution layers. Each channel was a concise CNN consisting of 6 layers of convolution. The addition element-wise operation was used to combine the two channels in order to add the features obtained from the first channel to the features obtained from the second channel. This channel summation was used so that the features extracted by the two channels complement each other based on the difference in scale. A 256 × 256 color image was used as the input for both CNN channels.

The first channel was a channel consisting of a convolution layer with a smaller filter size than the second channel. The first channel was preceded by a convolution layer consisting of 8 filters with each filter size of 7 × 7, while the second channel was preceded by a convolution layer with 8 filters measuring 15 × 15 each. The difference in filter sizes was intended to capture features at different scales and demonstrated that the first channel was responsible for extracting more detailed and local features, whereas the second channel extracted more global features. The size of the first and second channel filters were reduced in subsequent convolution layers to decrease the computational burden and the number of parameters involved. The details of the size of each layer of the two channels are shown in Table 1.

The difference in filter size between the two channels will extract features with varying sharpness, resulting in different feature maps. The feature map m extracted by each convolution layer using the F filter on the input tensor I is shown in Eq. 5.

$$m_{ij} = (I * F)_{ij} \tag{5}$$

The batch normalization layer, ReLU activation function, and the maxPooling layer followed the first to fifth convolution layers in both channels, while the maxPooling layer did not follow the sixth convolution layer. Stride [2 2] was used on all maxPooling layers and most convolution layers to reduce the size of the feature maps generated by these layers, decrease the number of parameters involved in computation, and speed up the training and classification tasks. Padding and stride [1 1] were used to keep the output size of the two channels the same so that addition operations could be performed at the ends of the two channels.

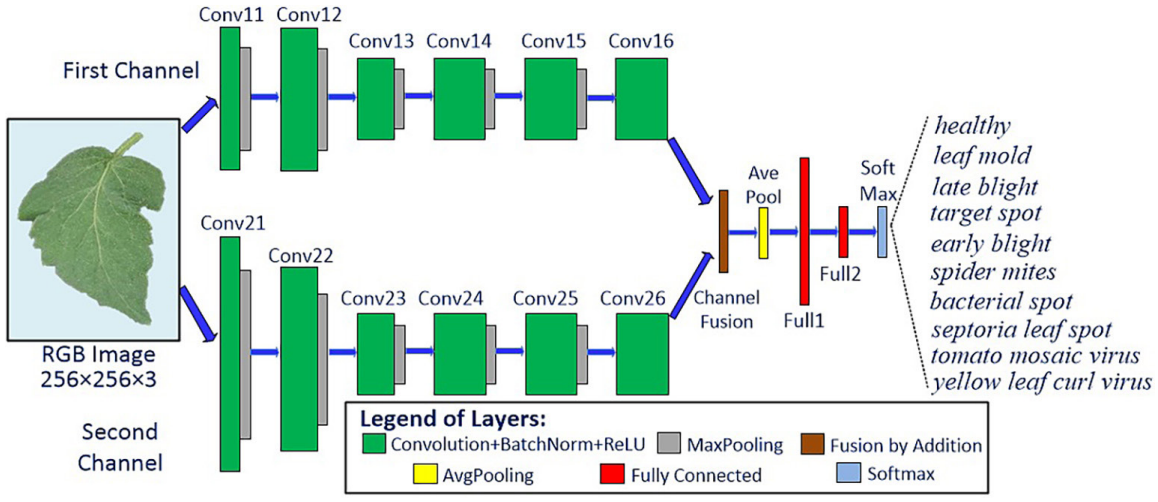

Figure 1. The architecture of the proposed concise CNN

Table 1. Details of each layer of the SlimPlantNet architecture

First Channel				Second Channel			
Layer Name	Type and Size	Output Size	Total Parameter	Layer Name	Type and Size	Output Size	Total Parameter
Conv 11	Convolution layer 8x7x7, stride [2 2], padding [2 2 2 2]	127× 127×8	1184	Conv 21	Convolution layer 8x15x15, stride [2 2], padding [2 2 2 2]	123× 123×8	5408
	BatchNorm Followed by ReLU	127× 127×8	16		BatchNorm Followed by ReLU	123× 123×8	16
	MaxPooling layer 7x7, stride [2 2], padding, same'	64×64×8	0		MaxPooling layer 3x3, stride [2 2], padding, same'	62×62×8	0
			0				0
Conv 12	Convolution layer 16x7x7, stride [2 2], padding [2 2 2 2]	31×31×16	6288	Conv 22	Convolution layer 16x9x9, stride [2 2], padding [2 2 2 2]	29×29×16	10384
	BatchNorm Followed by ReLU	31×31×16	32		BatchNorm Followed by ReLU	29×29×16	32
	MaxPooling layer 7x7, stride [2 2], padding, same'	16×16×16	0		MaxPooling layer 3x3, stride [2 2], padding, same'	15×15×16	0
			0				0
Conv 13	Convolution layer 32x3x3, stride [2 2], padding [2 2 2 2]	9×9×32	4640	Conv 23	Convolution layer 32×5×5, stride [2 2], padding [2 2 2 2]	8×8×32	12832
	BatchNorm Followed by ReLU	9×9×32	64		BatchNorm Followed by ReLU	8×8×32	64
	MaxPooling layer 3x3, stride [2 2], padding, same'	5×5×32	0		MaxPooling layer 3x3, stride [2 2], padding, same'	4×4×32	0
			0				0
Conv 14	Convolution layer 64x3x3, stride [2 2], padding [2 2 2 2]	4×4×64	18496	Conv 24	Convolution layer 64x3x3, stride [1 1], padding [1 1 1 1]	4×4×64	18496
	BatchNorm Followed by ReLU	4×4×64	128		BatchNorm Followed by ReLU	4×4×64	128
	MaxPooling layer 7x7, stride [2 2], padding ,same'	2×2×64	0		MaxPooling layer 3x3, stride [2 2], padding, same'	2×2×64	0
			0				0
Conv 15	Convolution layer 64x3x3, stride [2 2], padding [2 2 2 2]	2×2×64	36928	Conv 25	Convolution layer 64x3x3, stride [1 1], padding [1 1 1 1]	2×2×64	36928
	BatchNorm Followed by ReLU	2×2×64	128		BatchNorm Followed by ReLU	2×2×64	128
	MaxPooling layer 7x7, stride [2 2], padding, same'	1×1×64	0		MaxPooling layer 3x3, stride [2 2], padding, same'	1×1×64	0
			0				0
Conv 16	Convolution layer 64x3x3, stride [2 2], padding [1 1 1 1]	1×1×64	36928	Conv 26	Convolution layer 64x3x3, stride [1 1], padding [1 1 1 1]	1×1×64	36928
	BatchNorm Followed by ReLU	1×1×64	128		BatchNorm Followed by ReLU	1×1×64	128
Chan-nel Fusion	Fusion by Addition	1×1×64	-	Ave Pool	5x5 Average pooling, stride [2 2], padding, same'	1×1×64	-
	Fusion by Concatenation	1×1× 128	-				
Full1	Fully connected layer 1x64	1×1×64	4160	Full2	Fully connected layer 1x10	1×1×10	650

Transfer learning, training and testing

GoogleNet, MobileNetV2, and SqueezeNet were used to compare the performance of the concise CNN model proposed in this study. These three CNN modes were chosen because they have a small number of parameters compared to other CNNs and are frequently used in plant classification tasks. Transfer learning was performed in all three models using the PlantVillage dataset used

in this study. In this study, transfer learning was used to replace the classification layer in the three architectures with a new layer that was scaled to the number of classes in the dataset. All layers in the comparison architecture except the classification layer are preserved and frozen before being retrained only on the new layer using the PlantVillage dataset.

The proposed architecture named SlimPlantNet was trained from scratch. SlimPlantNet training and transfer learning comparison architecture were implemented in 20 epochs using the stochastic gradient descent with momentum (SGDM) optimization function. All comparison architectures used a learning rate of 0.0003, whereas the proposed architecture used a rate of 0.05. The test was performed after each epoch, and the classification time per image was performed sequentially after all training and testing was completed. The test was conducted on a computer with a Core i5 @ 2.7 GHz processor and 16 GB RAM memory with a Matlab 2019 environment.

Further investigation was carried out by examining the different effects of the fusion method of the two channels on SlimPlantNet via addition and concatenation operations. In Table 2, the architecture formed by the addition operation was named SlimPlantNet_addition, and the architecture formed by the concatenation operation was named SlimPlantNet_concat. The performance of each channel on the proposed architecture was also demonstrated as a single CNN architecture. This architecture was created by connecting each channel's sixth ReLU layer to the averagePooling layer and forming a complete CNN. SlimPlantNet_filter7 was the single CNN obtained from a channel with a 7 × 7 filter, and SlimPlantNet_filter15 was obtained from a channel with a 15 × 15 filter.

The performance of all CNN architectures involved in this study was measured using classification accuracy, loss, and average time required to classify each leaf image. Classification accuracy and loss are the most widely used parameters in testing the performance of a CNN (Maeda-Gutiérrez et al., 2020). Accuracy and loss were measured in training and testing every training epoch was completed, whereas the average classification time was measured during testing. Classification accuracy *CA* was measured based on the following Eq. 6:

$$CA = \frac{TruePos + TrueNeg}{TruePos + TrueNeg + \\ + FalPos + FalNeg} \tag{6}$$

where: $TruePos$ (true positive) is the number of positive images classified as positive;
$TrueNeg$ (true negative) is the number of negative images classified as negative;
$FalPos$ denotes the number of negative images classified as positive;
$FalNeg$ denotes the number of positive images classified as negative.

RESULTS AND DISCUSSION

Table 2 compares the performance of the proposed CNN model to the performance of the comparison CNN architectures in the classification of 14 different plant classes in the PlantVillage dataset, namely Apple healthy, Blueberry healthy, Cherry healthy, Corn (maize) healthy, Grape healthy, Orange Haunglongbing (Citrus greening), Peach healthy, Pepper healthy, Potato healthy, Raspberry healthy, Soybean healthy, Squash Powdery mildew, Strawberry healthy, and Tomato Healthy. The training was carried out in 20 epochs with the previously mentioned learning rate setting.

The test results show that the classification accuracy, as well as the loss value, are nearly identical between the proposed CNN model and the comparison CNN architecture. In this study, the accuracy of GoogleNet is almost identical to the accuracy of GoogleNet shown in (Maeda-Gutiérrez et al., 2020), which is 99.39% in 10 tomato classes from the PlantVillage dataset.

Significant differences can be seen in the training time and the required classification time per image. SlimPlantNet using addition and concatenation operations takes 0.0043 seconds to classify 1 image on the test computer used in this study, which is 5.12 times faster than SqueezeNet, 8.23 times than GoogleNet, and 9.40 times than MobileNetV2. The SlimPlantNet model also has a much shorter training time. Of course, the aspects of training time and classification time, as well as the number of parameters involved, will be the advantages of the SlimPlantNet model when implemented in an agricultural real-time monitoring system. Because there are fewer parameters, it can be implemented in computing devices with fewer resources and at a lower cost. The faster

Table 2. Performance testing of the proposed CNN model and comparison models on 14 PlantVillage dataset classes at 20 epochs

Architecture	Training time (hh:mm:ss)	Highest accuracy achieved (%)		Lowest loss achieved		Number of parameters	Classification time/image (s)
		Training	Testing	Training	Testing		
GoogleNet	11:36:09	100	99.87	0.0002	0.0044	6.8M	0.0354
SqueezeNet	05:11:05	100	99.92	0.0005	0.0038	1.24M	0.0220
MobileNetV2	20:50:16	100	99.80	0.0002	0.0081	3.4M	0.0404
SlimPlantNet_addition	01:44:22	100	99.58	0.0013	0.0201	222.99k	0.0043
SlimPlantNet_concat	01:33:28	100	99.60	0.0017	0.0198	234.09k	0.0043
SlimPlantNet_filter7	00:55:30	100	99.31	0.0018	0.0274	108.52k	0.0040
SlimPlantNet_filter15	00:56:11	100	99.23	0.0019	0.0269	128.10k	0.0029

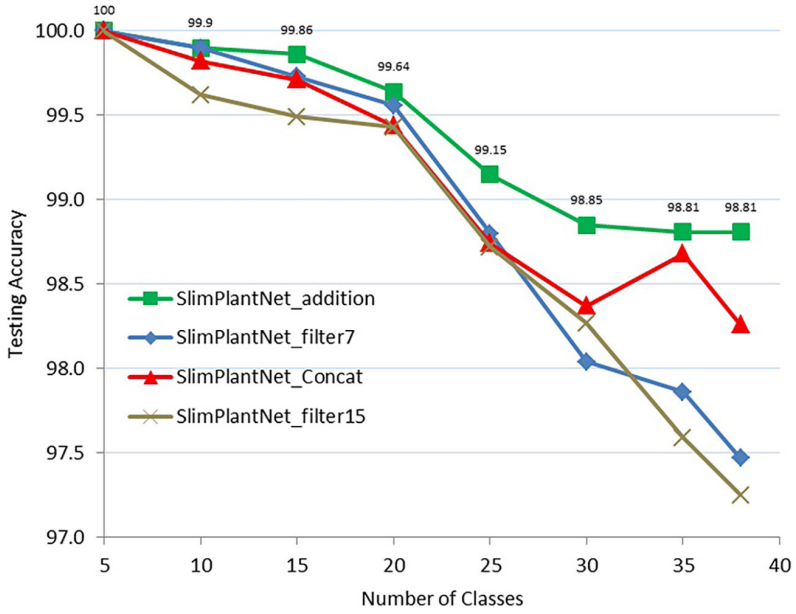

Figure 2. Performance comparison of SlimPlantNet combined using addition operation, concatenation operation, and performance of forming channels

classification time, the faster the monitoring system will run in real time.

This study also carried out further testing of the SlimPlanNet model using fusion with addition operations and concatenation operations based on the accuracy and loss of testing. SlimPlanNet's forming channels were also evaluated as a single CNN. This test was run in 60 epochs with varying numbers of classes, ranging from 5 to 38 classes, to see trends and performance limits in different data scales. Figure 2 depicts the test results.

SlimPlantNet addition achieved 98.81% classification accuracy on 38 classes and higher accuracy on a smaller number of classes in this test. In comparison, SqueezeNet achieved 98.46% test accuracy in (Liu et al., 2021) and 98.49% in (Aravind et al., 2020) in the classification of 38 PlantVillage dataset classes. In (Hassan et al., 2021), MobileNetV2 achieved a classification accuracy of 97.02% in the classification of 38 PlantVillage dataset classes. The study of (Sutaji and Rosyid, 2022) fine-tuned GoogleNet and MobileNetV2 by unfreezing the final convolution block layers. The accuracy of the test on the classification of 38 PlantVillage dataset classes was 98.34% for GoogleNet and 98.95% for MobileNetV2. The results of this test and comparisons show that SlimPlantNet addition's accuracy is quite reliable and competitive.

The results of this test also show that SlimPlantNet_addition is better in classifying the dataset than SlimPlantNet_concatenation and its constructor channel. SlimPlantNet_addition's accuracy is always higher, and the trend is quite stable as the number of classified classes increases.

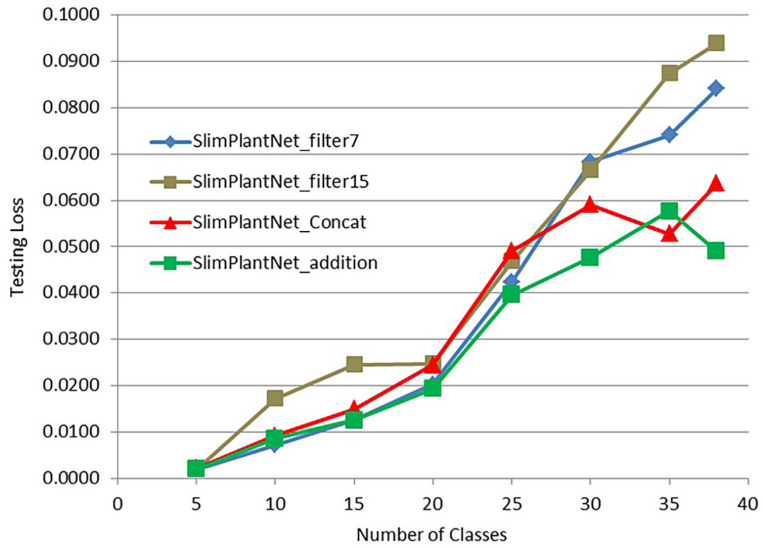

Figure 3. Test classification loss from the proposed model

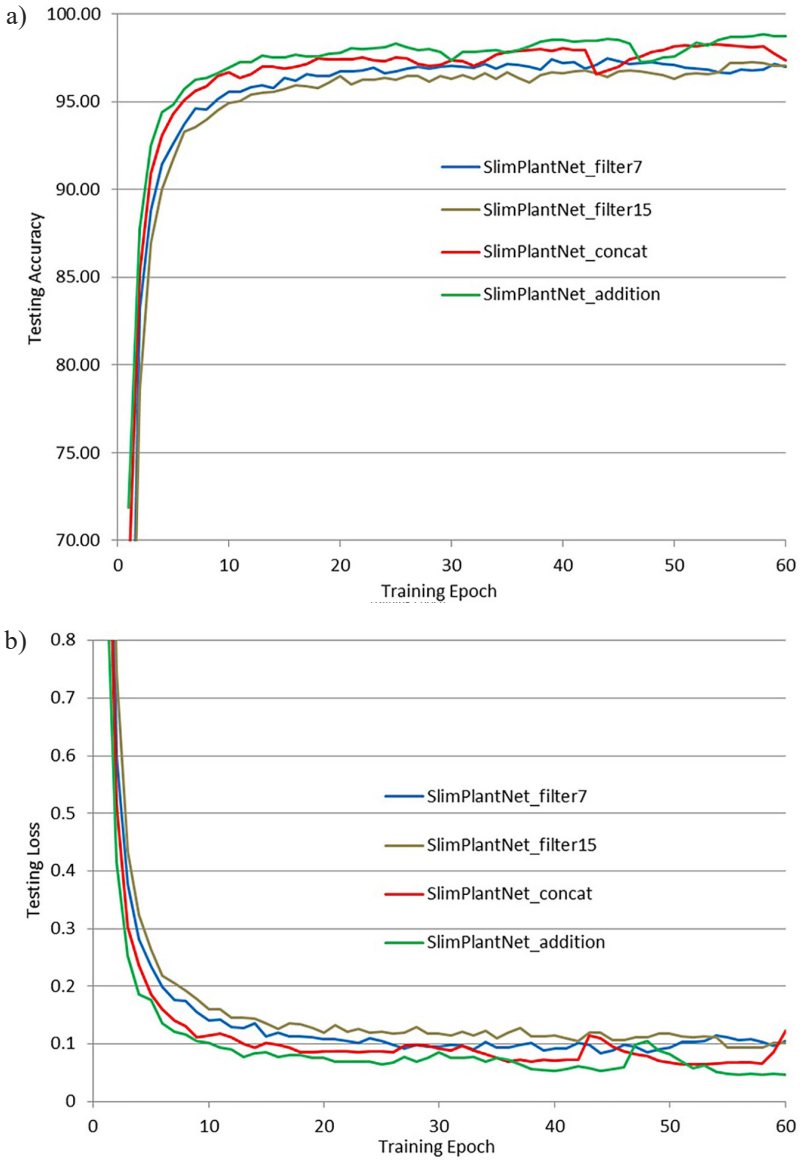

Figure 4. SlimPlantNet performance for plant classification of 38 classes in 60 epochs (a) Testing accuracy; b) Testing loss)

The performance comparison based on the test loss value shown in Figure 3 confirms the test results based on classification accuracy that the SlimPlantNet model with addition operation outperforms channel merging with concatenation operation. It is clear that the SlimPlantNet_addition loss value is always lower than the other three models.

The last comparison carried out in this study was a comparison of the trend of accuracy and test loss in 60 training epochs from SlimPlantNet, both combined with addition and concatenation operations as well as the SlimPlantNet channel. The training data on the PlantVillage dataset with 38 classes was highlighted, and the comparison results are shown in Figure 4. SlimPlantNet testing with addition operation fusion appears to be more accurate and stable over 60 training epochs, as does the loss value, which remains relatively low during the training period.

SlimPlanNet_filter7 and SlimPlanNet_filter15 have generally lower performance than SlimPlanNet_addition and SlimPlanNet_concat, but SlimPlanNet_filter7 outperforms SlimPlanNet_concat in classification from 5 to 25 dataset classes. Although the accuracy of SlimPlanNet_filter7 is slightly lower than that of SlimPlanNet_addition, if the number of parameters and classification speed are the most important factors, the SlimPlanNet_filter7 model can be recommended in classification tasks with fewer than 25 classes.

CONCLUSIONS

The results of the tests show that SlimPlantNet, which was formed by fusing two concise CNN channels using the addition operation, achieves reliable and competitive performance. Although classification speed and training time are not different, the SlimPlantNet model's classification accuracy and loss are better and more stable than the performance of SlimPlantNet one which uses channel fusion with concatenation operation. SlimPlantNet's classification accuracy on 14 different plant classes is nearly identical to the classification accuracy of the comparison architectures.

SlimPlantNet classifies images faster and with fewer parameters than comparison architectures. The speed in classification and the smaller number of parameters are the advantages of the SlimPlantNet model when implemented in a real-time monitoring system in the agricultural field with limited computer resources. Therefore, the future work after this study will be to integrate SlimPlantNet into the agricultural monitoring system. The use of the SlimPlantNet model in classification tasks other than plants is also interesting to investigate, particularly when the classification task involves a small number of classes, as in the PlantVillage dataset.

Acknowledgments

This work was supported by the Ministry of Research and Technology/National Research and Innovation Agency of Republic Indonesia (Kementerian Riset Dan Teknologi/Badan Riset Dan Inovasi Nasional Republik Indonesia) under the scheme of Penelitian Disertasi Doktor (PDD) 2021 with contract number of 56/SP2H/LT/DPRM/2021.

REFERENCES

1. Abade A., Ferreira P.A., Vidal F.B. 2021. Plant diseases recognition on images using convolutional neural networks: A systematic review. Computers and Electronics in Agriculture, 185(July 2020). DOI: 10.1016/j.compag.2021.106125
2. Aravind K.R., Maheswari P., Raja P., Szczepański C. 2020. Crop disease classification using deep learning approach: an overview and a case study. Deep Learning for Data Analytics, 173–195. DOI: 10.1016/b978-0-12-819764-6.00010-7
3. Awais M., Iqbal M.T.B, Bae S.H. 2021. Revisiting Internal Covariate Shift for Batch Normalization. IEEE Transactions on Neural Networks and Learning Systems, 32(11), 5082–5092. DOI: 10.1109/TNNLS.2020.3026784
4. Bjorck J., Gomes C., Selman B., Weinberger K.Q. 2018. Understanding batch normalization. Advances in Neural Information Processing Systems, 2018-Decem(NeurIPS), 7694–7705.
5. Brahimi M., Kamel B., Moussaoui A. 2017. Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. In Applied Artificial Intelligence. DOI: 10.1080/08839514.2017.1315516
6. Brahimi M. 2018. Deep learning for plants diseases. Springer International Publishing. DOI: 10.1007/978-3-319-90403-0
7. Chouhan S.S., Kaul A., Singh U.P., Jain S. 2018. Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology. IEEE Access, 6(i), 8852–8863. DOI: 10.1109/ACCESS.2018.2800685

8. Elhassouny A., Smarandache F. 2019. Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks. 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), 1–4. DOI: 10.1109/ICCSRE.2019.8807737
9. Ferentinos K.P. 2018. Deep learning models for plant disease detection and diagnosis. In Computers and Electronics in Agriculture, 145. DOI: 10.1016/j.compag.2018.01.009
10. Hassan S.M., Maji A.K., Jasiński M., Leonowicz Z., Jasińska E. 2021. Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach. Electronics (Switzerland), 10(12). DOI: 10.3390/electronics10121388
11. Howlader M.R., Habiba U., Faisal R.H., Rahman M.M. 2019. Automatic Recognition of Guava Leaf Diseases using Deep Convolution Neural Network. 2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019, 1–5. DOI: 10.1109/ECACE.2019.8679421
12. Hughes D.P., Salathe M. 2015. An open access repository of images on plant health to enable the development of mobile disease diagnostics. http://arxiv.org/abs/1511.08060
13. Iandola F.N., Han S., Moskewicz M.W., Ashraf K., Dally W.J., Keutzer K. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, 1–13. http://arxiv.org/abs/1602.07360
14. Ioffe S., Szegedy C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (F. Bach and D. Blei (eds.); PMLR, 37, 448–456). http://proceedings.mlr.press/v37/ioffe15.pdf
15. Kamilaris A., Prenafeta-Boldú F.X. 2018. Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147(July 2017), 70–90. DOI: 10.1016/j.compag.2018.02.016
16. Karthik R., Hariharan M., Anand S., Mathikshara P., Johnson A., Menaka R. 2020. Attention embedded residual CNN for disease detection in tomato leaves. Applied Soft Computing Journal, 86, 105933. DOI: 10.1016/j.asoc.2019.105933
17. Kaur S., Pandey S., Goel S. 2018. Semi-automatic leaf disease detection and classification system for soybean culture. IET Image Processing, 12(6), 1038–1048. DOI: 10.1049/iet-ipr.2017.0822
18. Knoll F.J., Czymmek V., Poczihoski S., Holtorf T., Hussmann S. 2018. Improving efficiency of organic farming by using a deep learning classification approach. Computers and Electronics in Agriculture, 153, 347–356. DOI: 10.1016/j.compag.2018.08.032
19. Lajoie-O'Malley A., Bronson K., Burg S.V.D, Klerkx L. 2020. The future(s) of digital agriculture and sustainable food systems: An analysis of high-level policy documents. Ecosystem Services, 45(August), 101183. DOI: 10.1016/j.ecoser.2020.101183
20. Liu Y., Gao G., Zhang Z. 2021. Plant disease detection based on lightweight CNN model. 2021 4th International Conference on Information and Computer Technologies (ICICT), 64–68. DOI: 10.1109/ICICT52872.2021.00018
21. Lu J., Hu J., Zhao G., Mei F., Zhang C. 2017. An In-field Automatic Wheat Disease Diagnosis System. In Computers and Electronics in Agriculture, 142. DOI: 10.1016/j.compag.2017.09.012
22. Luna R.G.D., Dadios E.P., Bandala A.A. 2019. Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition. IEEE Region 10 Annual International Conference, Proceedings/TENCON, 2018-Octob(October), 1414–1419. DOI: 10.1109/TENCON.2018.8650088
23. Ma J., Du K., Zheng F., Zhang L., Gong Z., Sun Z. 2018. A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. Computers and Electronics in Agriculture, 154, 18–24. DOI: 10.1016/j.compag.2018.08.048
24. Maeda-Gutiérrez V., Galván-Tejada C.E., Zanella-Calzada L.A., Celaya-Padilla J.M., Galván-Tejada J.I., Gamboa-Rosales H., Luna-García H., Magallanes-Quintanar R., Guerrero-Méndez C.A., Olvera-Olvera C.A. 2020. Comparison of convolutional neural network architectures for classification of tomato plant diseases. In Applied Sciences (Switzerland), 10(4). DOI: 10.3390/app10041245
25. McCool C., Perez T., Upcroft B. 2017. Mixtures of Lightweight Deep Convolutional Neural Networks: Applied to Agricultural Robotics. IEEE Robotics and Automation Letters, 2(3), 1344–1351. DOI: 10.1109/LRA.2017.2667039
26. Mohanty S.P., Hughes D.P., Salathé M. 2016. Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7(September), 1–10. DOI: 10.3389/fpls.2016.01419
27. Mokhtar U., Ali M.A.S., Hassanien A.E., Hefny H. 2015. Identifying Two of Tomatoes Leaf Viruses Using Support Vector Machine. Advances in Intelligent Systems and Computing, 339, 771–782. DOI: 10.1007/978-81-322-2250-7
28. Najdenovska E., Dutoit F., Tran D., Plummer C., Wallbridge N., Camps C., Raileanu L.E. 2021. Classification of Plant Electrophysiology Signals for Detection of Spider Mites Infestation in Tomatoes. In Applied Sciences, 11(4). DOI: 10.3390/app11041414
29. Rahman C.R., Arko P.S., Ali M.E., Khan M.A.I, Apon S.H., Nowrin F., Wasif A. 2020. Identification and recognition of rice diseases and pests using convolutional neural networks. Biosystems

Engineering, 194, 112–120. DOI: 10.1016/j.biosystemseng.2020.03.020

30. Rangarajan A.K., Purushothaman R., Ramesh A. 2018. Tomato crop disease classification using pre-trained deep learning algorithm. Procedia Computer Science, 133, 1040–1047. DOI: 10.1016/j.procs.2018.07.070
31. Saleem M.H., Potgieter J., Arif K.M. 2020. Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers. In Plants, 9(10). DOI: 10.3390/plants9101319
32. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 4510–4520. DOI: 10.1109/CVPR.2018.00474
33. Santurkar S., Tsipras D., Ilyas A., Madry A. 2018. How does batch normalization help optimization? Advances in Neural Information Processing Systems, 2018-Decem(NeurIPS), 2483–2493.
34. Suryawati E., Sustika R., Yuwana R.S., Subekti A., Pardede H.F. 2018. Deep Structured Convolutional Neural Network for Tomato Diseases Detection. 2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 385–390. DOI: 10.1109/ICACSIS.2018.8618169
35. Sutaji D., Rosyid H. 2022. Convolutional Neural Network (CNN) Models for Crop Diseases Classification. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 4(2). DOI: 10.22219/kinetik.v7i2.1443
36. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D. 2015. Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1002/jctb.4820
37. Too E.C., Yujian L., Njuki S., Yingchun L. 2019. A comparative study of fine-tuning deep learning models for plant disease identification. Computers and Electronics in Agriculture, 161(October 2017), 272–279. DOI: 10.1016/j.compag.2018.03.032
38. Wu Q., Wang F. 2019. Concatenate convolutional neural networks for non-intrusive load monitoring across complex background. Energies, 12(8). DOI: 10.3390/en12081572
39. Wu Y., Wang Z., Shi Y., Hu J. 2020. Enabling On-Device CNN Training by Self-Supervised Instance Filtering and Error Map Pruning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(11), 3445–3457. DOI: 10.1109/TCAD.2020.3012216
40. Yamashita R., Nishio M., Do R.K.G., Togashi K. 2018. Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition. Smart Innovation, Systems and Technologies. DOI: 10.1007/978-981-15-7078-0_3
41. Zhang S., Huang W., Zhang C. 2019. Three-channel convolutional neural networks for vegetable leaf disease recognition. Cognitive Systems Research, 53, 31–41. DOI: 10.1016/j.cogsys.2018.04.006