Sensors and Materials, Vol. 33, No. 11 (2021) 4031–4044
MYU Tokyo

S & M 2743

EfficientNet: A Low-bandwidth IoT Image Sensor Framework for Cassava Leaf Disease Classification

Chih-Cheng Chen,[1,4] Ju Yan Ba,[2] Tie Jun Li,[2] Christopher Chun Ki Chan,[3*] Kun Ching Wang,[5**] and Zhen Liu[2]

[1]Department of Automatic Control Engineering, Feng Chia University, Taichung 40724, Taiwan
[2]School of Ocean Information Engineering, Jimei University, Fujian Province, Xiamen 361021, China
[3]Department of Information Management, Chaoyang University of Technology, Taichung 413310, Taiwan
[4]Department of Aeronautical Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan
[5]Department of Mechanical and Computer-Aided Engineering, Feng Chia University, Taichung 40724, Taiwan

(Received July 1, 2021; accepted October 4, 2021; online published November 5, 2021)

Keywords: machine vision, leaf diseases, image enhancement, TTA enhancement

Following the cassava leaf disease classification process, we successfully design a novel convolutional neural network (CNN) framework called EfficientNet using low-bandwidth image sensors and a combination of image enhancement and image classification methods. For this study, we have employed low-bandwidth, small-scale IoT image sensors in a farm to capture images of cassava leaves at periodic intervals. We employ data enhancement techniques such as test-time augmentation (TTA) and cutmix, cutout, and *k*-fold to accurately classify and evaluate the pathology of cassava plants. We carry out multiple simulated experiments to classify and evaluate diseases found in five cassava leaf datasets. Our framework is capable of producing relatively accurate classification results despite small differences between the test set images, and we achieved a classification accuracy for our final test set of 89%, comparable with that in similar studies. Experimental results obtained using a sensor show that EfficientNet significantly outperforms a state-of-the-art cassava leaf disease classification model.

1. Introduction

Cassava is a common plant in southern Africa and is the second largest source of food carbohydrates produced in small-scale farming. More than 80% of family farms grow the cassava crop, making it one of the major staple foods in the tropics. Cassava is subject to a few common viral diseases that cause the crop to develop toxins that may cause serious harm or death when consumed. Computer vision techniques can visually identify common viral diseases using sensors and allow farmers to precisely treat certain diseases and increase yields. It is difficult to detect virus-infested cassava through visual inspection through human eyes, and doing so often results in low accuracy rates and a high labor cost. A sensor-based crop inspection system for developing countries must be designed under the constraint of footage from low-bandwidth cameras as developing countries have low access and/or no need for high-definition/

[*]Corresponding author: e-mail: christopherckchan@cyut.edu.tw
[**]Corresponding author: e-mail: kunching.wang@gmail.com
https://doi.org/10.18494/SAM.2021.3526

high-bandwidth cameras for farming. A low-bandwidth camera sensor is a type of low-energy CMOS sensor with reasonably low power consumption that follows low-bandwidth sensor network IEEE 802.1 protocols, and IoT image sensors are employed in various locations around a greenhouse farm to capture images of cassava leaves.

A computer's interpretation of an image is through pixel information, which is different from a human's interpretation of an image through the human visual system. Detection via computerized interpretation works by transforming an image to pixels, which have information and can be input to a computer in a specific way, such as a vector, then processed in parallel or series. Utilizing pixel-by-pixel information with mathematical operations such as convolution can produce a computerized abstraction that produces unique 'features'. These features are then categorized into bins that are useful for performing a task. In industry and academia, there are five types of image classification: (1) image classification based on the simultaneous interpretation of different sensor information; (2) image classification based on the properties of training samples; (3) image classification based on various parameters used in data; (4) image classification based on certain properties of pixels; and (5) image classification based on spatial information.

Object detection determines the position of a target object in an image, whereas target detection is the identification of a target object in an image of any natural scene. Target detection is divided into three stages: image region selection, feature extraction, and image classification.

It is important to also understand edge extraction technology for computer vision as it is important for our enhancement methodology. In computer vision, an object's edge contains a signal mutation, an irregular structure, and instability. Thus, objects in an image can be interpreted via a signal that exists in an object's contour. By determining the position of an image edge and its signal, we can obtain a sample that can be used for image recognition. Performing signal processing and edge detection well can (1) improve image quality; (2) provide information for the separation of targets; (3) provide information on a scene; (4) provide information for further feature recognition. Edge extraction technology has made rapid progress, increasing the accuracy of classification.

The dataset used in our study consists of more than 20000 photos of cassava plants, which were captured in Uganda. These cassava plants were identified and classified into four disease types and healthy plants by the National Crop Resources Institute. We posit that, with the help of EfficientNet, farmers can quickly identify the type of disease and treat the cassava plant more effectively.

In this study, we designed a convolutional neural network (CNN) framework called EfficientNet to classify cassava leaf diseases with high accuracy. We briefly introduce CNNs in Sect. 2, as well as give a short introduction to pooling and fully connected layers. In Sect. 3, we discuss related works in cassava leaf disease detection. Section 4 shows the structure and principle of data enhancement and label smoothing in detail. In Sect. 5, we describe the principle and hierarchical structure of EfficientNet in detail. In Sect. 6, we describe our results through experiments, in which we discuss the *k*-fold and test-time augmentation (TTA) strategies for test set training, and our process for verifying the accuracy of our method. We conclude our study in Sect. 7.

2. Convolutional Neural Networks

CNNs are widely used in image classification, knowledge mapping, recommendation systems, target detection, target recognition, image recognition, speech recognition, and other fields. After Hubel and Wiesel explored the cat's visual nerve, they proposed the concept of a receptive field(1) and further discovered that there is an association between the visual cortex of the brain and the processing and classification of external input. They explained the concept of the sensory field; visual input spaces and cells contribute to the whole visual cortex and can be mimicked to realize accurate computer vision techniques. CNNs(2) are then created from these concepts and can be divided into three parts. The first part is the image input layer, the second part is the convolutional and pooling layers for feature extraction, and the third part is a joining layer, which decomposes and merges image features into a column vector.

2.1 Convolutional layer

A convolutional layer(3) has an optimized backpropagation solution, and together these form a convolution unit. A convolution layer comprises multiple convolutional units. A convolution operation determines low-level features of an image, such as edges, lines, and faces. With each additional convolution layer, a network can extract more complex features from low-level features. The recognition of objects is essentially performed by mathematical reasoning and abstraction from the features of points and edges. For example, in the first layer, the information regarding points and edges is extracted from pixels and used as the input for the next layer. In the second layer, rectangles, ellipses, and other information are recognized from the points and edges, and used as the input for the next layer, and so forth. This process occurs until a level of mathematical abstraction that is sufficient for the relevant matching task is obtained. Because many features can be extracted, one can adjust the number of convolutions and features to extract according to the desired purpose. Figure 1 shows the convolution process for two matrices, where the pink matrix is the resulting matrix.

In Fig. 1, the size of the input picture is 5×5. After convolution with the convolution core of size 3×3, the original input space is mapped to a 3×3 area. Here, the convolution operation extracts the features of the input image, so that every pixel in the image can be learned in this way.

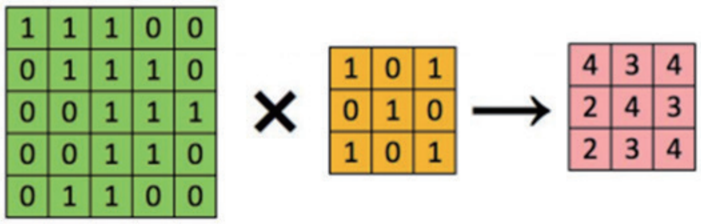


Fig. 1. (Color online) Convolution process for CNNs.

2.2 Pooling layer

We obtain abstract features through the preset convolution layer, which are prone to overfitting.(4) Therefore, it is necessary to reduce the dimension of features by downsampling in the pooling layer. The maximum or average pooling method is used to obtain the maximum or average value in a fixed area to reduce the dimension of features. This method can not only preserve image features but also prevent overfitting. The maximum pooling process is shown in Fig. 2.

2.3 Fully connected layer

The function of the fully connected layer is to connect all the features in the form of column vectors from the 2D feature map output by the front layer. The output value is sent to the classifier, and the process is shown in Fig. 3.

The convolution and pooling layers map the original data to a feature space, which means that the accuracy of the feature matrix and the abstraction of the original data are increased. To input this information into a classifier, we convert the feature space into a column vector, and the next layer maps the convolution and pooling layer outputs to a linear separable space. The fully connected layer can obtain high-order features from the convolution and pooling operations, simplify the parameter model, and reduce the numbers of neurons and training parameters.(4)

3. Related Works on Cassava Leaf Disease Detection

3.1 Deep models for cassava leaf disease detection

According to our comprehensive survey on cassava leaf disease detection methods using image processing and artificial intelligence, current methodologies use techniques such as artificial intelligence, IoT, algorithms based on rules, machine learning regression techniques, image processing, transfer learning, hyperspectral imagery, leaf extraction, and segmentation.(5)

There exist deep learning models that require substantial computational resources and significant postprocessing time to obtain high recognition rates. Studies have been reported on

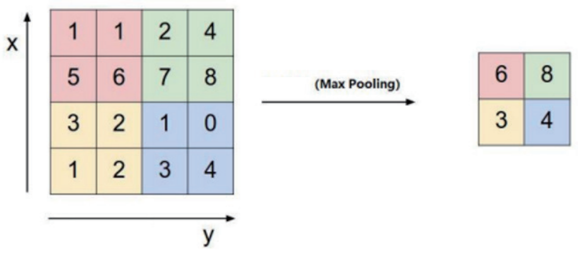


Fig. 2. (Color online) Maximum pooling process.

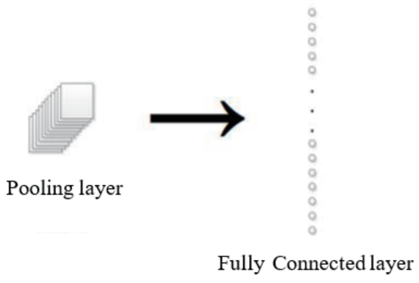


Fig. 3. Fully connected layer.

pest damage recognition utilizing deep transfer learning (93% classification accuracy rate),(6) the resolution of class imbalances using a synthetic minority oversampling technique for underrepresented classes,(7) and a deep learning model with no augmentation processes (85% accuracy rate).(8) Other related studies include those on deep residual CNNs with distinct block processing(9) and the real-time embedded system MobileNetV2 deep learning model (65.6% accuracy rate).(10) Preliminary cassava leaf disease detection was performed with a cubic support vector machine (SVM) model and a coarse Gaussian SVM, which yielded accuracies of 83.9 and 61.6%, respectively.(11)

3.2 Lightweight models for cassava leaf disease detection

Lightweight models, similar to our framework, which are more practical for developing countries, include a mobile implementation using a single-shot detector called MobileNet to identify foliar symptoms of cassava leaf diseases(12) and a recognition model using low-quality images that are augmented with enhanced data techniques.(13)

The mobile implementation(12) requires sufficient training examples from diverse data sources to improve its detection accuracy and better capture the diversity of cassava leaf diseases that occur in the real world. The methodology using low-quality images(13) attempts to solve the recognition problem by using a novel image color histogram transformation technique to generate synthetic images, which is based on the convolution of Chebyshev orthogonal functions with the probability distribution functions of image color histograms. Our method is similar in that we use color transformations for normalization but do not rely on color histograms.

4. Preprocessing of Training Data

4.1 Applied data enhancement techniques

Data enhancement technology refers to reusing and learning each image through processing without increasing the number of images in the expanded dataset.(14) In our study, we employed key data enhancement techniques to achieve an effect similar to that of increasing the number of images in the expanded dataset. A geometric transformation of the image is carried out as shown in Fig. 4. As shown in Fig. 5, we applied a color transformation to the image.(15)

4.2 Label smoothing

In this paper, we utilize label smoothing, which is a regularization method(16) that improves generalization and model calibration. Smoothing the labels prevents the network from being overconfident and overfitting results after training. In the loss function, where the model applies the softmax function to the penultimate layer's logit vectors to compute its output probabilities, the gradient is bounded between 0 and 1. These boundaries are too far apart and we adjusted them to 0.3 and 0.7, so that gradient descent and smoothed labels will compute better probabilities with small logit gaps. Doing so will partly address the problem of poor generalization.

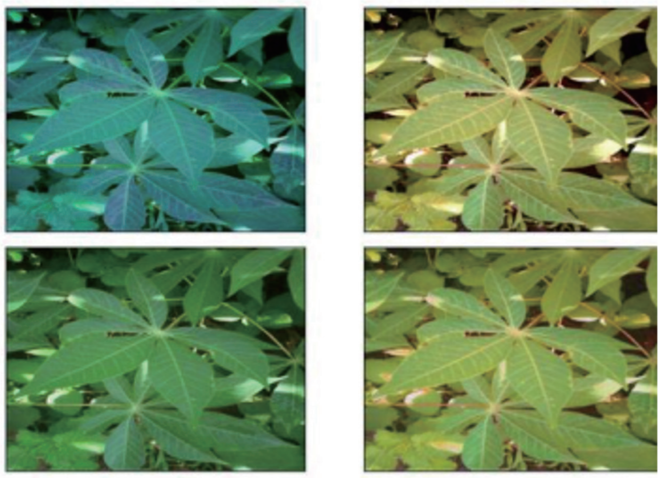


Fig. 4. (Color online) Geometric transformation of the image.

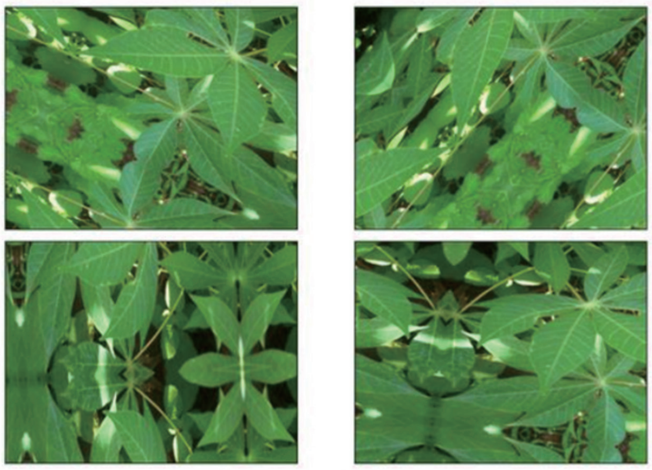


Fig. 5. (Color online) Color transformation.

In the classification problem, there are generally two values of tags in the training data, 0 and 1, where 0 represents a certain category and 1 represents the opposite category. Thus, in traditional one-hot coding, the label vector y_i is

$$y_i = \begin{cases} 1, & i = target, \\ 0, & i \neq target. \end{cases} \tag{1}$$

When the model is deployed for training, its minimum loss function is

$$H(y,p) = -\sum_{i}^{K} y_i \log p_i, \tag{2}$$

where parameters p_i correspond to the output logic vector z of the penultimate layer in the model, which can be calculated by the softmax function

$$p_i = -\frac{\exp(z_i)}{\sum_j^K \exp(z_i)}. \tag{3}$$

For the past models, the aim was for the target tag value predicted at the end to be close to 1 and the non-target tag value to be close to 0. However, if the logic vector is too large, the network will be overfitted and the generalization ability will be poor. Therefore, the cross-entropy function is not necessarily the best choice as the objective function.

Label smoothing combines a uniform distribution with updated label vectors,i.e.,

$$\hat{y}_i = y_{hot}(1-a) + a/k, \tag{4}$$

where k is the total number of categories and a is a small superparameter (usually 0.1), determined as

$$\hat{y}_i = \begin{cases} 1-a, \ i = target, \\ a/K, \ i \neq target. \end{cases} \tag{5}$$

Label smoothing can be understood as adding interference noise to the label distribution of the original real dataset. In this way, the overfitting of the model for labels can be avoided and the difference between labels is not too large, which also improves the generalization ability. Cassava leaf classification can benefit from label smoothing as it changes the representations learned by the penultimate layer of the network by encouraging the representations of training examples from the same class to group in tight clusters.

4.3 Mixup, cutout, and cutmix

For our study, we also applied three preprocessing methods to the data to improve the accuracy of our results. Cassava leaves captured by low-resolution sensors are often subject to noise and environmental interference, and by utilizing simple preprocessing methods, we improve the model accuracy without adding additional data. The three methods are mixup, cutout, and cutmix.

Mixup: Mix any two samples according to the desired proportion. The results of the classification are distributed according to the proportion.

Cutout: Arbitrarily delete part of the region in the sample and fill the region with a pixel value of 0. The result of classification remains unchanged.

Cutmix: Delete part of an image and randomly fill the part with nonzero pixel values, which are allocated according to a certain proportion. These functions are illustrated in Fig. 6.

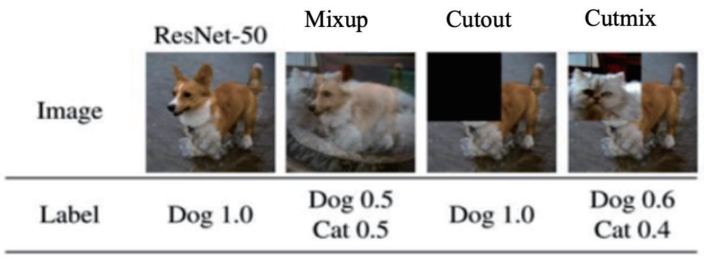


Fig. 6. (Color online) Illustration of the mixup, cutout, and cutmix functions.

5. EfficientNet Algorithm

5.1 Convolutional neural network model

The depth and width of the model network and the resolution of the input image correspond to the number of layers, the channel size, and the input tensor size in each stage, respectively. Because the network depth and width and the resolution are adjusted continuously, the accuracy of the model tends to be stable. The general framework of the EfficientNet model is shown in Fig. 7.

As each image in the dataset passes through the neural network, it obtains a feature map, similarly to that in encoding, as shown in Fig. 8. Each dimension vector of the feature map can be understood as the feature vector corresponding to part of the original image, namely, a local feature. Finally, the features in the decoding part are input to the fully connected layer then classified by softmax. The principle of the EfficientNet model is shown in Fig. 9.

The EfficientNet network layer size and channel number are shown in Table 1. Through the coefficients h, W, and R, EfficientNet can be extended to other EfficientNet networks, as shown in Table 2.

5.2 Depthwise separable convolution

The depth can be separated by convolution shown in Fig. 10, which can reduce the number of convolution parameters. The parameter quantity shown in Fig. 11 is $3 \times 3 \times 10 \times 4$, that is, 360 parameters are required. EfficientNet has seven stages, each of which has the same structure but different superparameters.

The number of parameters required is $3 \times 3 \times 10 + 1 \times 1 \times 10 \times 6 = 150$. After processing by a single 1×1 convolution kernel, the features of these channels can be combined linearly, so that the features are related to each other. A diagram of depthwise separable convolution is shown in Fig. 11.

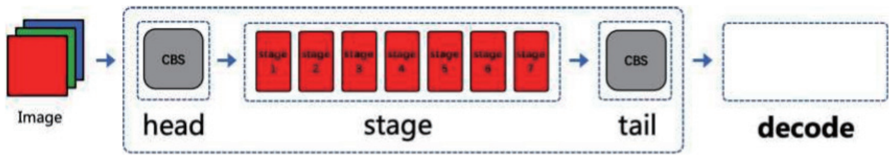


Fig. 7. (Color online) Framework of EfficientNet model.

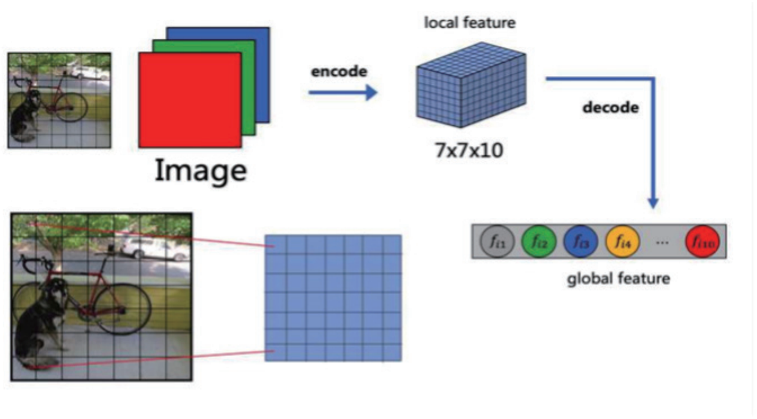


Fig. 8. (Color online) Schematic diagram of encoding principle.

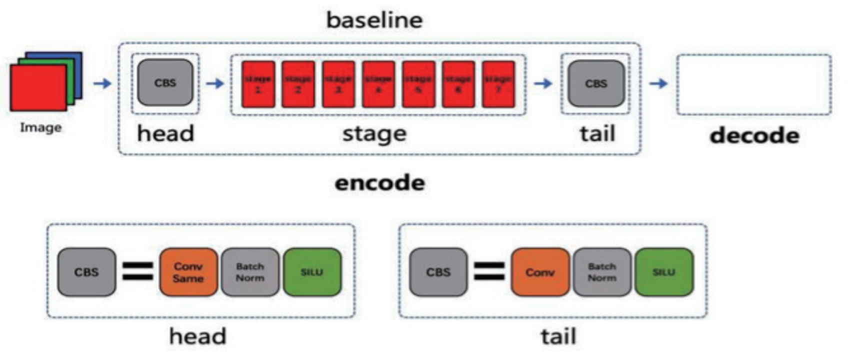


Fig. 9. (Color online) Schematic of EfficientNet principle.

Table 1
Size and number of channels of each network layer of EfficientNet.

Stage	Operator	Resolution	Channels	Layers
1	Conv3 × 3	224 × 224	32	1
2	MBConVt k3 × 3	112 × 112	16	1
3	MBConV6, k3 × 3	112 × 112	24	2
4	MBConV6, k5 × 5	56 × 56	40	2
5	MBCOnV6, k3 × 3	28 × 28	80	3
6	MBeOnV6, k5 × 5	14 × 14	112	3
7	MBConV6, k5 × 5	14 × 14	192	4
8	MBConV6$_t$ k3 × 3	7 × 7	320	1
9	ConVIXI & PooIing & FC	7 × 7	1280	1

Table 2
Parameter values of different EfficientNet versions.

	Width	Depth	Resolution
EfficientNet-b0	1.0	1.0	224
EfficientNet-b1	1.0	1.1	240
EfficientNet-b2	1.1	1.1	240
EfficientNet-b3	1.2	1.4	300
EfficientNet-b4	1.4	1.8	380
EfficientNet-b5	1.6	2.2	456
EfficientNet-b6	1.8	2.6	528
EfficientNet-b7	2.0	3.1	600

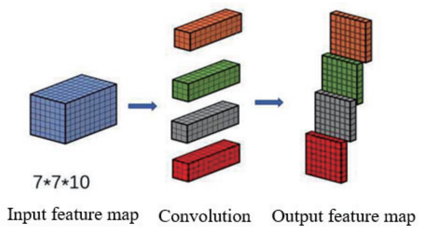


Fig. 10. (Color online) Diagram of ordinary convolution.

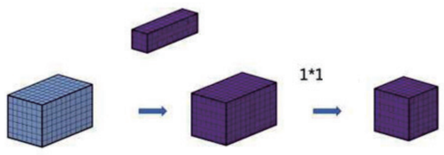


Fig. 11. (Color online) Diagram of depthwise separable convolution.

6. Experiment and Analysis

6.1 Setting experimental parameters for *k*-fold and TTA

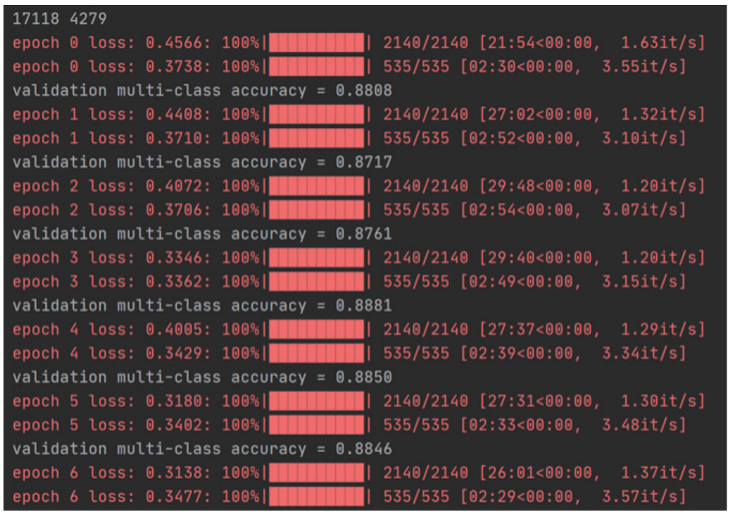
There are four functions of data enhancement. The first is to avoid overfitting. When a dataset has some special feature preferences, such as the pictures in the dataset are taken from the same natural scene, data enhancement methods such as cutout and cutmix are used to prevent the model from learning information unrelated to the target. The second is to improve the robustness of the model and reduce its sensitivity to the data. When the training data encounters special situations, such as occlusion, brightness, and fuzziness, it is easy to identify errors. The model data can be improved by adding noise and concealment. The third is to increase the amount of training data and improve the generalization ability of the model. The fourth is to avoid sample imbalance. In industrial defect detection, there is often a large difference between the numbers of positive and negative samples. For example, in the training samples, the number of good samples is far greater than that of poor samples, which results in the network being more inclined to learn the characteristics of good samples than those of poor samples, reducing the effectiveness of the network. By using data enhancement and modification techniques, the information shown in the data can be enhanced.

6.2 Evaluation of image classification accuracy

With *k* set to 5 in the *k*-fold method, the image set used for testing is separated from the image set used to train the model network parameters. The accuracy of the test results is acceptable. In the test, the TTA parameter is set to 3, that is, one test image is judged three times. The results for the first six batches are shown in Fig. 12 and the results for the last five batches are shown in Fig. 13. The results of the test set are shown in Fig. 14. In the 12 rounds of deep learning training, the loss rate of the verification set is as low as 0.27 and the correct rate is 89.73% in the final test set, which is not involved in the training.

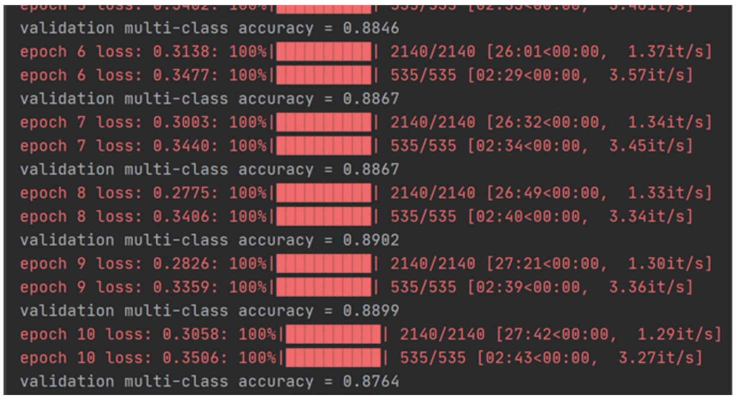
6.3 Experimental comparison

The datasets with and without data enhancement were trained and tested. The experimental results obtained using a low-bandwidth camera sensor are shown in Figs. 15 and 16. After five



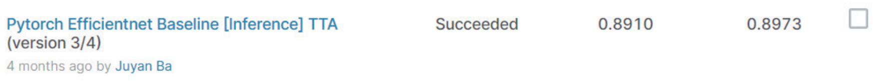
```
17118 4279
epoch 0 loss: 0.4566: 100%|██████████| 2140/2140 [21:54<00:00,  1.63it/s]
epoch 0 loss: 0.3738: 100%|██████████| 535/535 [02:30<00:00,  3.55it/s]
validation multi-class accuracy = 0.8808
epoch 1 loss: 0.4408: 100%|██████████| 2140/2140 [27:02<00:00,  1.32it/s]
epoch 1 loss: 0.3710: 100%|██████████| 535/535 [02:52<00:00,  3.10it/s]
validation multi-class accuracy = 0.8717
epoch 2 loss: 0.4072: 100%|██████████| 2140/2140 [29:48<00:00,  1.20it/s]
epoch 2 loss: 0.3706: 100%|██████████| 535/535 [02:54<00:00,  3.07it/s]
validation multi-class accuracy = 0.8761
epoch 3 loss: 0.3346: 100%|██████████| 2140/2140 [29:40<00:00,  1.20it/s]
epoch 3 loss: 0.3362: 100%|██████████| 535/535 [02:49<00:00,  3.15it/s]
validation multi-class accuracy = 0.8881
epoch 4 loss: 0.4005: 100%|██████████| 2140/2140 [27:37<00:00,  1.29it/s]
epoch 4 loss: 0.3429: 100%|██████████| 535/535 [02:39<00:00,  3.34it/s]
validation multi-class accuracy = 0.8850
epoch 5 loss: 0.3180: 100%|██████████| 2140/2140 [27:31<00:00,  1.30it/s]
epoch 5 loss: 0.3402: 100%|██████████| 535/535 [02:33<00:00,  3.48it/s]
validation multi-class accuracy = 0.8846
epoch 6 loss: 0.3138: 100%|██████████| 2140/2140 [26:01<00:00,  1.37it/s]
epoch 6 loss: 0.3477: 100%|██████████| 535/535 [02:29<00:00,  3.57it/s]
```

Fig. 12. (Color online) Results for the first six batches.



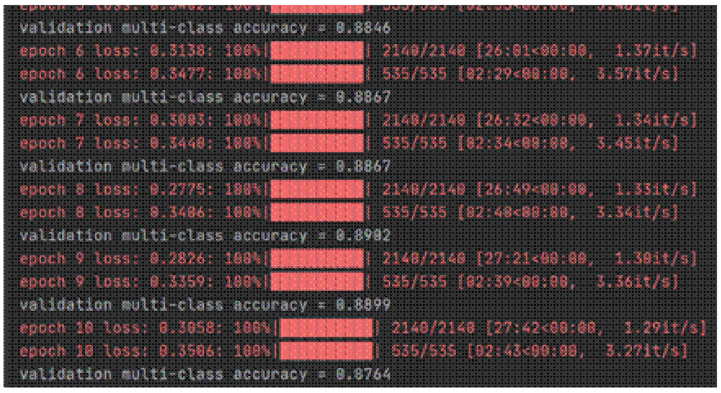
```
validation multi-class accuracy = 0.8846
epoch 6 loss: 0.3138: 100%|██████████| 2140/2140 [26:01<00:00,  1.37it/s]
epoch 6 loss: 0.3477: 100%|██████████| 535/535 [02:29<00:00,  3.57it/s]
validation multi-class accuracy = 0.8867
epoch 7 loss: 0.3003: 100%|██████████| 2140/2140 [26:32<00:00,  1.34it/s]
epoch 7 loss: 0.3440: 100%|██████████| 535/535 [02:34<00:00,  3.45it/s]
validation multi-class accuracy = 0.8867
epoch 8 loss: 0.2775: 100%|██████████| 2140/2140 [26:49<00:00,  1.33it/s]
epoch 8 loss: 0.3406: 100%|██████████| 535/535 [02:40<00:00,  3.34it/s]
validation multi-class accuracy = 0.8902
epoch 9 loss: 0.2826: 100%|██████████| 2140/2140 [27:21<00:00,  1.30it/s]
epoch 9 loss: 0.3359: 100%|██████████| 535/535 [02:39<00:00,  3.36it/s]
validation multi-class accuracy = 0.8899
epoch 10 loss: 0.3058: 100%|██████████| 2140/2140 [27:42<00:00,  1.29it/s]
epoch 10 loss: 0.3506: 100%|██████████| 535/535 [02:43<00:00,  3.27it/s]
validation multi-class accuracy = 0.8764
```

Fig. 13. (Color online) Results for the last five batches.



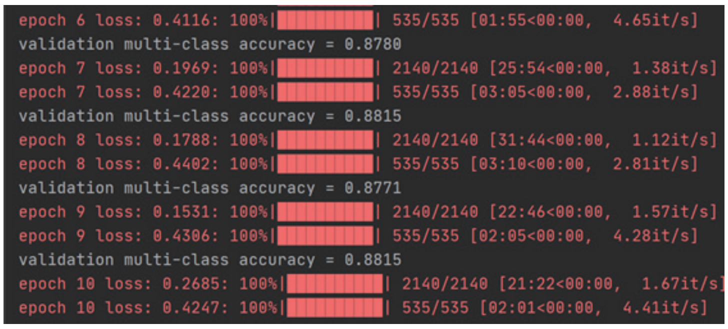
Pytorch Efficientnet Baseline [Inference] TTA (version 3/4) Succeeded 0.8910 0.8973
4 months ago by Juyan Ba

Fig. 14. (Color online) Test set results.



```
validation multi-class accuracy = 0.8846
epoch 6 loss: 0.3138: 100%|██████████| 2140/2140 [26:01<00:00,  1.37it/s]
epoch 6 loss: 0.3477: 100%|██████████| 535/535 [02:29<00:00,  3.57it/s]
validation multi-class accuracy = 0.8867
epoch 7 loss: 0.3003: 100%|██████████| 2140/2140 [26:32<00:00,  1.34it/s]
epoch 7 loss: 0.3440: 100%|██████████| 535/535 [02:34<00:00,  3.45it/s]
validation multi-class accuracy = 0.8867
epoch 8 loss: 0.2775: 100%|██████████| 2140/2140 [26:49<00:00,  1.33it/s]
epoch 8 loss: 0.3406: 100%|██████████| 535/535 [02:40<00:00,  3.34it/s]
validation multi-class accuracy = 0.8902
epoch 9 loss: 0.2826: 100%|██████████| 2140/2140 [27:21<00:00,  1.30it/s]
epoch 9 loss: 0.3359: 100%|██████████| 535/535 [02:39<00:00,  3.36it/s]
validation multi-class accuracy = 0.8899
epoch 10 loss: 0.3058: 100%|██████████| 2140/2140 [27:42<00:00,  1.29it/s]
epoch 10 loss: 0.3506: 100%|██████████| 535/535 [02:43<00:00,  3.27it/s]
validation multi-class accuracy = 0.8764
```

Fig. 15. (Color online) Training results of the last five batches without data enhancement training.



```
epoch 6 loss: 0.4116: 100%|██████████| 535/535 [01:55<00:00,  4.65it/s]
validation multi-class accuracy = 0.8780
epoch 7 loss: 0.1969: 100%|██████████| 2140/2140 [25:54<00:00,  1.38it/s]
epoch 7 loss: 0.4220: 100%|██████████| 535/535 [03:05<00:00,  2.88it/s]
validation multi-class accuracy = 0.8815
epoch 8 loss: 0.1788: 100%|██████████| 2140/2140 [31:44<00:00,  1.12it/s]
epoch 8 loss: 0.4402: 100%|██████████| 535/535 [03:10<00:00,  2.81it/s]
validation multi-class accuracy = 0.8771
epoch 9 loss: 0.1531: 100%|██████████| 2140/2140 [22:46<00:00,  1.57it/s]
epoch 9 loss: 0.4306: 100%|██████████| 535/535 [02:05<00:00,  4.28it/s]
validation multi-class accuracy = 0.8815
epoch 10 loss: 0.2685: 100%|██████████| 2140/2140 [21:22<00:00,  1.67it/s]
epoch 10 loss: 0.4247: 100%|██████████| 535/535 [02:01<00:00,  4.41it/s]
```

Fig. 16. (Color online) Training results of the last five batches after data enhancement training.

training epochs, it can be seen that in the 10th batch of training, the loss is reduced to 0.26 in the training and 0.42 in the testing. However, in the 10th batch of training with data enhancement, the loss is reduced to 0.38 in the training and 0.35 in the testing. We can see that the training with data enhancement significantly reduces the loss compared with the training without data enhancement, and that data enhancement can reduce the image loss in testing. Without data enhancement training, the generalization ability is poor and the performance of the training set is overfitted.

7. Conclusion

We developed a Cassava leaf classification neural network called EfficientNet to classify diseases from images of cassava leaves. In our proposed classification process, we enhance a specific cassava leaf dataset, but our framework can theoretically be applied to other cassava leaf datasets in future work. We set up IoT image sensors at various locations in a greenhouse farm containing cassava leaves to capture closeup images of cassava leaves. We utilized data enhancement techniques to improve the generalization ability of the network and provide higher recognition rates for cassava leaves in natural scenes. However, performance characteristics may differ according to environmental conditions, such as natural light and fog (which can cause colors in the image to be imbalanced), or the camera capability. When images captured by a camera are used, the details of leaves cannot be seen because of the low resolution of the images;

therefore, it is important to employ novel postprocessing techniques for images taken in natural scenes.

Acknowledgments

This work was supported by the Natural Science Foundation of Fujian Province, China (Grant No. 2020J01714, 2020J01713, and MOST 110-2622-E-035-006).

References

1 D. H. Hubel and T. N. Wiesel: J. Physiol. **160** (1962) 106. https://doi.org/10.1113/jphysiol.1962.sp006837
2 H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers: IEEE Trans. Med. Imaging **35** (2016) 1285. https://doi.org/10.1109/TMI.2016.2528162
3 S. Albawi, T. A. Mohammed, and S. Al-Zawi: IEEE ICET (2017) 1. https://doi.org/10.1109/ICEngTechnol.2017.8308186
4 K. O’Shea and R. Nash: arXiv (2015) 1511.08458. https://arxiv.org/abs/1511.08458v2
5 H. Parikshith, S. N. Rajath, and S. P. Kumar: ICCVBIC (2019) 304. https://doi.org/10.1007/978-3-030-37218-7_35
6 A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes: Front. Plant Sci. **8** (2017) 1852. https://doi.org/10.3389/fpls.2017.01852
7 G. Sambasivam and G. D. Opiyo: Egypt. Inform. J. **22** (2021) 27. https://doi.org/10.1016/j.eij.2020.02.007
8 R. Surya and E. Gautama: IEEE ICSITech (2020) 97. https://doi.org/10.1109/ICSITech49800.2020.9392051
9 D. O. Oyewola, E. G. Dada, S. Misra, and R. Damaševičius: PeerJ Comput. Sci. **7** (2021) e352. https://doi.org/10.7717/peerj-cs.352
10 H. R. Ayu, A. Surtono, and D. K. Apriyanto: J. Phys. Conf. **1751** (2021) 012072. https://doi.org/10.1088/1742-6596/1751/1/012072
11 O. Emuoyibofarhe, J. O. Emuoyibofarhe, S. Adebayo, A. Ayandiji, O. Demeji, and O. James: IJCSSE **8** (2019) 166. https://www.researchgate.net/publication/336020567_Detection_and_Classification_of_Cassava_Diseases_Using_Machine_Learning
12 A. Ramcharan, P. McCloskey, K. Baranowski, N. Mbilinyi, L. Mrisho, M. Ndalahwa, J. Legg, and D. P. Hughes: Front. Plant Sci. **10** (2019) 272. https://doi.org/10.3389/fpls.2019.00272
13 O. O. Abayomi-Alli, R. Damaševičius, S. Misra, and R. Maskeliūnas: Expert Syst. **38** (2021) e12746. https://doi.org/10.1111/exsy.12746
14 E. Mwebaze, T. Gebru, A. Frome, S. Nsumba, and J. Tusubira: arXiv (2019) 1908.02900. https://arxiv.org/abs/1908.02900v2
15 M. C. Wilson, A. M. Mutka, A. W. Hummel, J. Berry, R. D. Chauhan, A. Vijayaraghavan, N. J. Taylor, D. F. Voytas, D. H. Chitwood, and R. S. Bart: New Phytol. **213** (2017) 1632. https://doi.org/10.1111/nph.14443
16 C. B. Zhang, P. T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M. M. Cheng: IEEE Trans. Image Process **30** (2021) 5984. https://doi.org/10.1109/TIP.2021.3089942

About the Authors

Chih Cheng Chen has been a professor at Jimei University, China, since 2017. He became a member of IEEE in 2011 and a senior member in 2016. He earned his M.S. degree in 2005 and recently earned his Ph.D. degree from the Department of Mechatronics Engineering, National Changhua University of Education. He has published 43 academic articles and owns three patents. He has also implemented several projects funded by the National Natural Science Foundation of China, Fujian Province. His research interests include AIoT technology, machine learning, information security, and RFID applications. (chenccheng@fcu.edu.tw)

Ju Yan Ba has been a student at Jimei University, China, since 2017. He earned his C.E. degree in 2021 and is continuing his study in computer science toward earning a master's degree. (896729202@qq.com)

Tie Jun Li has been an associate professor at Jimei University, China, since 2015. He received his B.S. degree from Chongqing University, China, in 2000 and his M.S. and Ph.D. degrees from Chongqing University in 2005 and 2018, respectively. He has published 21 academic articles and owns 11 patents. He also implemented several projects supported by the Natural Science Foundation of China, Fujian Province. His research interests are in machine learning, terahertz technology, and sensors. (litiejun@jmu.edu.cn)

Christopher Chun Ki Chan has been an assistant professor at Chaoyang University of Technology, Taiwan, since 2019. He is an IEEE member and an active reviewer for computer science journals and conferences such as TEMS. He earned his M.S. and Ph.D. degrees from the Department of Computer Science, Ryerson University, Canada, where he specialized in deep reinforcement learning, robotics, and Industry 4.0. (christopherckchan@cyut.edu.tw)

Kun Ching Wang is a professor of the Department of Mechanical and Computer-Aided Engineering, Feng Chia University, Taiwan. He became a member of IEEE in 2012 and a senior member in 2019. He received his Ph.D. degree in control engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 2005. He has published over 40 journal and conference papers. His research interests include speech/audio processing and recognition, machine learning, computer vision, advanced driver assistance systems, and IoT application.

Zhen Liu is currently studying for a master's degree at Jimei University, China. His research interests include deep learning, signal processing, image recognition, and bearing fault diagnosis.