Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 53 (2019) 31–41

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Three-channel convolutional neural networks for vegetable leaf disease recognition

Shanwen Zhang [a], Wenzhun Huang [a], Chuanlei Zhang [b,*]

[a] College of Information Engineering, Xijing University, Xi'an 710123, China
[b] Tianjin University of Science and Technology, Tianjin 300222, China

Received 19 November 2017; received in revised form 11 April 2018; accepted 19 April 2018
Available online 5 May 2018

Abstract

The color information of diseased leaf is the main basis for leaf based plant disease recognition. To make use of color information, a novel three-channel convolutional neural networks (TCCNN) model is constructed by combining three color components for vegetable leaf disease recognition. In the model, each channel of TCCNN is fed by one of three color components of RGB diseased leaf image, the convolutional feature in each CNN is learned and transmitted to the next convolutional layer and pooling layer in turn, then the features are fused through a fully connected fusion layer to get a deep-level disease recognition feature vector. Finally, a softmax layer makes use of the feature vector to classify the input images into the predefined classes. The proposed method can automatically learn the representative features from the complex diseased leaf images, and effectively recognize vegetable diseases. The experimental results validate that the proposed method outperforms the state-of-the-art methods of the vegetable leaf disease recognition.

Keywords: Vegetable leaf disease recognition; Feature extraction and selection; Convolutional neural networks (CNN); Three-channel CNN (TCCNN)

1. Introduction

Vegetables are often threatened by various kinds of diseases. Leaf based vegetable leaf disease recognition is always an important research topic in many fields such as ecology, pattern recognition and image processing, and many methods have been proposed for vegetable leaf disease recognition (Barbedo, 2016). However, the performance of these methods is unsatisfactory because of the limited discriminative description ability of low-level features extracted from diseased leaf images. It is known that the disease recognition rates of the traditional approaches rely heavily on the lesion segmentation and hand-designed features that require expensive work and expert knowledge (Wang & Huang, 2009; Wang, Huang, & Xu, 2010). To improve disease recognition rate, many researchers tried to use various algorithms to extract large various features from each diseased leaf image, such as seven invariant moments, SIFT (Scale-invariant feature transform), Gabor transform, global-local singular value and sparse representation (Guo, Hastie, & Tibshirani, 2007; Zhang & Wang, 2016; Zhang, Wu, You, et al., 2017). By the existing leaf based disease recognition methods, it is easy to extract over 100 kinds of features from only one diseased leaf image, but the contributions of the different features to the disease recognition are very different, and it is difficult to determine which features are optimal and robust for disease recognition. Especially, if the background is complex with other leaves or plants, the diseased leaf image segmentation may be questionable. Most

* Corresponding author.
E-mail addresses: wjdw716@163.com (S. Zhang), 97313114@tust.edu.cn (C. Zhang).

https://doi.org/10.1016/j.cogsys.2018.04.006
1389-0417/

methods fail to effectively segment the leaf and corresponding lesion image from its background which will lead to unreliable disease recognition results. By now, leaf based vegetable disease recognition is still a challenge research because of the very complexity of diseased leaf image, as shown in Fig. 1. From Fig. 1, it can be seen that three leaf images of the same kind disease (i.e., Blotch) and their lesions vary a lot from each other (i.e., Gray leaf spot) in lighting, brightness, scale, position and background.

Many feature extraction and selection, dimensionality reduction (Huang & Jiang, 2012), subspace learning (Li & Huang, 2008; Li, Huang, Du, et al., 2006), sparse representation algorithms (Guo et al., 2007; Wright, Yang, Ganesh, et al., 2009) and various neural networks (Huang, 1996, 1999; Huang & Du, 2008; Huang, Huang, Yuan, et al., 2017; Huang, Ip, & Chi, 2004; Zhao, Huang, & Sun, 2004; Zhao, Huang, & Guo, 2003) and classifiers (Bao, Chen, & Wang, 2014; Bao, Wang, Chen, et al., 2017; Huang, 2004; Huang, Ip, Law, et al., 2005; Huang & Du, 2008; Liu & Huang, 2008) can be applied to disease recognition, but it is difficult to segment diseased leaf image and extract robust features for disease recognition. Recently, deep learning has been widely applied to image classification, computer vision and pattern recognition applications, and has achieved state-of the-art performance (Huang & Zhang, 2017; Zhang & Zhang, 2017; Schmidhuber, 2015). Deep learning can automatically learn the high-level characteristics from the massive original complex images, instead of a lot of image preprocessing, lesion segmentation, and artificially-designed feature extraction. Deep learning has been applied to plant species recognition and plant disease recognition. Mohanty, Hughes, and Marcel (2016) trained a deep learning model for recognizing 14 crop species and 26 crop diseases. The trained model achieves an accuracy of 99.35% on a held-out test set. CNNs can performs both feature extraction and image classification in the same architecture, and outperform the latest leaf based plant disease recognition methods (Wiatowski & Bölcskei, 2016). Srdjan, Marko, Andras, et al. (2016) proposed a plant disease recognition approach based on CNNs to distinguish between healthy leaves and 13 different diseases. Amara, Bouaziz, and Algergawy (2017) proposed a banana disease recognition approach based on LeNet architecture. The results demonstrate the effectiveness of the proposed approach even under the challenging conditions. The presence of a large amount of vegetable diseased leaf images and powerful computing infrastructure makes CNN be a suitable candidate for the disease recognition.

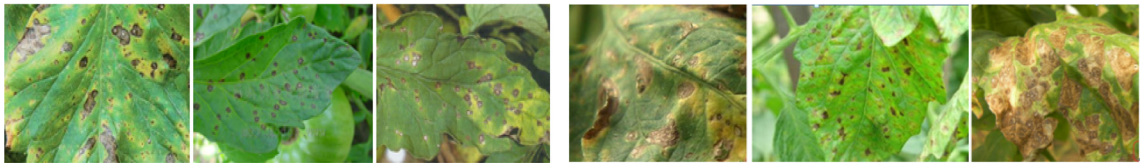
The color characteristics of the diseased leaf image can be described its three color components of R, G and B. Because the actual plant diseased leaf images are complex with miscellaneous background, as shown in Fig. 1, the color information provided by a single color component is limited and the extracted features by the classical disease recognition methods are often one-sided and incomplete. Representing the disease class by the diseased leaf image through multi-color-components can somewhat alleviate this problem. Inspired by the recent developments of plant disease recognition and multi-column CNN (He & Tian, 2016; Schmidhuber, 2012), a TCCNN model is proposed for vegetable leaf disease recognition by integrating three different color components of the diseased leaf image. Its main contributions are as follows:

- A TCCNN model is constructed for vegetable leaf disease recognition.
- The model requires minimal image preprocessing process, and omits the process of diseased leaf image segmentation and hand-crafted feature extraction.
- The model can automatically learn to extract the high-level classifying features directly from the color diseased leaf image, and then the recognition rate will be greatly improved.

The rest of the paper is organized as follows: in Section 2, the architecture and technical details of CNNs are introduced. A vegetable leaf disease recognition approach based on TCCNN is proposed in Section 3. Experimental results are reported in Section 4. Section 5 concludes the paper and provides the future work.

2. Convolutional neural networks (CNN)

The general architecture of CNN is shown in Fig. 2, including an input layer, three convolutional layers (C1, C2, C3), two pooling layers (P1, P2), two fully-connected layers (FC) and an output layer (Wiatowski & Bölcskei, 2016), where the convolutional and pooling layers act as feature descriptors, the fully-connected and output layers act as a classifier, and the neurons between adjacent layers are not fully-connected but partially connected. CNN is



(A) Blotch (B) Gray leaf spot

Fig. 1. Tomato diseased leaf images.

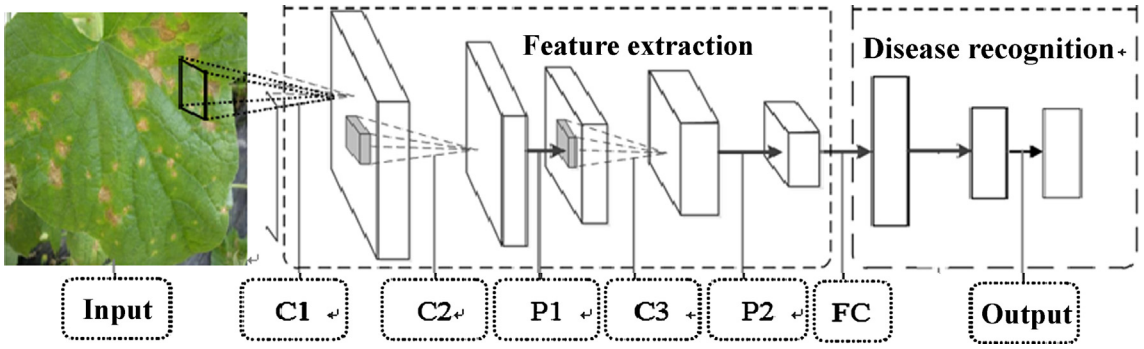


Fig. 2. Typical CNNs architecture.

regarded as a black-box classifier. Its framework, kernel, receptive field, the number of layers and the number of feature maps of each layer are five main parameters that affect its applicability. CNN has been widely applied to practical tasks and achieved better results without knowing the working process. In the section, we introduce the basic components of CNN in detail.

- Convolutional layer

Convolutional layer is an important component of CNN, consisting of a set of learnable filters (or kernels). The raw pixels of the input image are convoluted with each filter through the dot product between the kernel pixel and the input pixel. Each pixel in the input image represents a neuron, and each neuron in the convolutional layer is locally connected with its local neurons. The weight sharing scheme is used to significantly control the free parameter number, and drastically reduce the complexity and training computation load of CNNs (He & Tian, 2016).

In the lth convolutional layer, the output of the jth convolutional filter is convolved with a set of weights, which is calculated by summing up the contributions (weighted by the filter components) from the previous layer neurons:

$$z_j^l = \sum_{i \in M_j} x_i^{l-1} * W_{ij}^l + b_j^l \tag{1}$$

where l is network layer, j notes convolutional kernel, '*' is convolutional operator, z_j^l is the output of the ith neuron of the lth convolutional layer, M_j is the set of input maps in the (j-1)th receptive field, x_i^{l-1} is the ith input feature in M_j of the lth convolutional layer, W_{ij}^l is the ith weight of the jth convolutional filter in the lth layer, b_j^l is the ith additive bias in the lth layer for the output layer.

Then, the convolutional layer applies the nonlinearity of z_i^l:

$$x_j^l = f(z_j^l) \tag{2}$$

where f is an activation function.

The activation function is used to learn the intrinsical features from the original image. There are several kinds of activation functions, such as ReLu (Rectified Linear Units), Logistic-Sigmoid and Hyperbolic tangent function, where ReLu is widely used in deep learning applications. It is defined as

$$f(z_i^l) = \max(0, z_i^l) \tag{3}$$

- Pooling layer

Pooling layer often follow the convolutional layer. The output of the convolutional layer is the input of the pooling layer. The pooling operation is used to reduce the spatial size of the feature representation and computational cost in CNNs, so as to quickly obtain the spatial invariant features. Suppose the input layer is a $N \times N$ image, the output is a $N/q \times N/q$ feature map by the pooling operation over a lot of receptive fields of size of $q \times q$, where each $q \times q$ receptive field is reduced to just a single value. In the lth pooling layer, the output feature on the jth local receptive field can be calculated as follows:

$$x_j^l = g(\gamma_j^l down(x_j^{l-1}) + b_j^l) \tag{4}$$

where g is a pooling function, $down(*)$ is the pooling operator, and γ is a weight.

There are three common pooling operations: (1) max-pooling, that is done by applying a max-filter to non-overlapping sub-regions of the last layer representation, and simply selecting the maximum value among the elements in the same sub-region; (2) average-pooling, that likes the max-pooling, outputs the average of each sub-region; and (3) random-pooling, that randomly selects a value in each local receptive field in accordance with the probability of this value. By a random pooling operation, the neurons that are not the largest inspirit in the feature plane can be also used to avoid the overfitting of CNNs (Wiatowski & Bölcskei, 2016).

- Fully-connected layer

After convolutional layers and pooling layers, there are one or several fully-connected layers, and the high-level reasoning in CNN is carried on the fully-connected layers. The characteristics of the input image are extracted using the convolutional and pooling operators, and then are input into the first fully-connected layer, and the output of the last fully-connected layer is fed to the output layer for

image recognition and classification (Mohanty et al., 2016). Each neuron in the fully-connected layer is fully-connected to all neurons of its previous layer. Let the input of the fully-connected layer be x with the size of $k \times l$, which results in a matrix $Q_{k\times l}$, then the output of CNN is as follows,

$$M(x) = g(Q * x) \tag{5}$$

where g is an activation function, which is often chosen as an identity function.

- Output layer

The output layer is also called classification layer. The output layer with a Softmax function makes use of the feature vector of the last fully-connected layer to classify the original images into the predefined classes. The error is propagated back over a Softmax layer. The input of Softmax is a feature vector resulted from the learning process, and its output is a probability vector to indicate that an image belongs to a given class:

$$p_j^{(i)} = \exp(W_j^T x^{(i)} + a_j) \Big/ \sum_{n=1}^{C} \exp(W_n^T x^{(i)} + a_n) \tag{6}$$

where $p_j^{(i)}$ is the probability of the sample $x^{(i)}$ belonging to the jth class image, W_j and a_j are the jth classifier parameter, and C is the number of the predicted image classes.

- Training process

CNN is a multi-layer feed-forward neural network to learn features from the input images. CNN can be trained only by using a known pattern without an accurate mathematical model between any input and output. Before training CNN, all parameters of CNN need to be initialized with a lot of different small random numbers. CNN is often trained by back-propagation neural network (BPNN). The training process of CNN and traditional neural network (NN) is basically the same, including two stages:

(1) Forward propagation. Take a sample from the sample set and input to CNN, and calculate the corresponding output. At this stage, the feature information is extracted from the input layer to transmit to the next layer step by step, and the output in the last layer is multiplied by the weight matrix of each layer to obtain the final classification results.
(2) Back propagation. Calculate the difference between the actual output and the corresponding ideal output; then adjust the weight matrix by inverse propagation error. According to the forward propagation, the weight w_j^l and biases b_j^l of the jth characteristic in the lth layer are updated by,

$$x_j^l = f(u_j^l), \quad u_j^l = w_j^l x_j^{l-1} + b_j^l \tag{7}$$

where $f(.)$ is also an activation function.

In Eq. (7), w_j^l and b_j^l can be calculated using the back propagation strategy, that is to say, spread the $(l+1)$th layer error to the lth layer, and continue to spread until to the first convolutional layer, and then calculate the derivative between error and weight. Then all parameters of each layer can be updated by iterating two partial derivatives of the error when the error is minimized. Finally the CNN training process is completed.

The stochastic gradient descent algorithm is employed to update the parameters (including weights and biases) using a subset of the training set so as to minimize the loss function and learn the optimal parameter set (W, b). The key problem is to find the weight loss function $E(W, b)$ of the partial derivative parameters. So, for the specific image classification task, selecting a suitable loss function is more important. The parameters (W, b) can be updated by,

$$w_{ij}^{l+1} = w_{ij}^l - \alpha \frac{\partial E(W,b)}{\partial w_j^l}, \quad b_i^{l+1} = b_i^l - \alpha \frac{\partial E(W,b)}{\partial b_j^l} \tag{8}$$

In Eq. (7), the parameters of the ith neuron of the $(l+1)$th layer and the jth neuron of the lth layer represent the biases term of the ith neuron in the $(l+1)$th layer, and α is a learning rate.

In training process, the greater α the training rate, the more probable the network falls into the local optimal; on the contrary, if the smaller α, the slower the training process, the optimal solution may be found. Therefore, α should be chosen according to the actual application.

3. The proposed method for vegetable leaf disease recognition

For fact CNN based recognition tasks, network depth, number of feature planes, size of convolutional kernel and moving step of CNN should be properly set up. Because the color characteristic of the diseased leaf is very important for recognizing plant diseases, by making use of the color characteristics of the diseased leaf image, a TCCNN is proposed for vegetable leaf disease recognition by integrating the different components of color diseased leaf image.

3.1. Architecture of TCCNN

The architecture of TCCNN is shown in Fig. 3A, including an input layer, three color channels, three CNNs, a fully-connected fusion layer, a Softmax layer and an output layer. Considering the amount of training diseased leaf image available, the architecture of each CNN is set as shown in Fig. 3B, including four convolution layers (C1, C2, C3, C4), three pooling layers (P1, P2, P3), and two fully-connected layers (FC1, FC2). Dropout regularization is used in the fully-connected layer and ReLU activation is applied to each convolutional and fully-connected layers. The input of each channel is a specific color component.

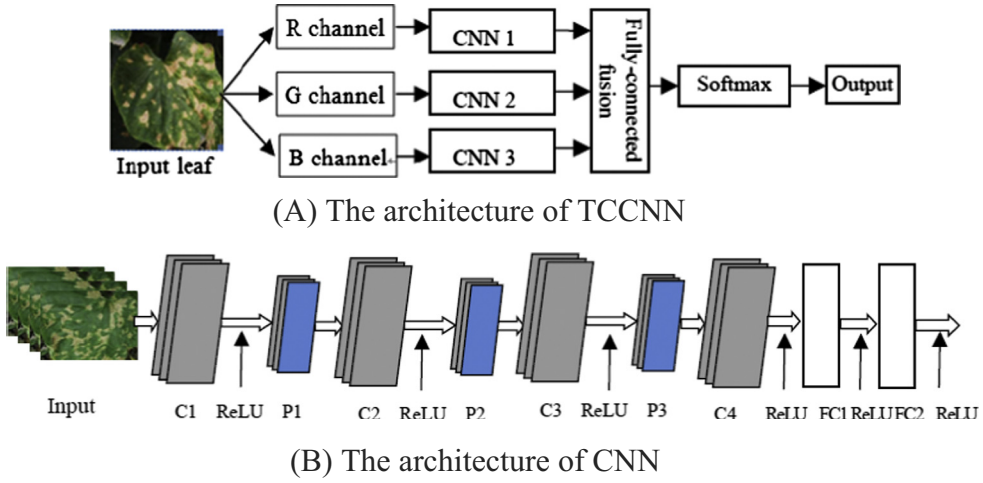


Fig. 3. Vegetable leaf disease recognition process based on TCCNN.

Three components are applied here, namely, R, G and B components of the color diseased leaf image.

As shown in Fig. 3A, each diseased leaf image is firstly divided into three components and input into three CNN respectively. The input of each channel is the different components of the diseased leaf image, and the feature extraction process from three components is independent in three channels. So the TCCNN can be extended to a multi-channel CNN for more than three components. The different components for the initialized input image are decided manually. In TCCNN, the architecture of each CNN can be selected as the same as LeNet5, AlexNet and ConvNet (Schmidhuber, 2012; Wiatowski & Bölcskei, 2016).

In TCCNN, although each CNN have the same architecture and the original parameters, it has its own independent weights, and the final error is obtained with the output of three CNN, so that all layers act as a bias for each other. In each CNN, every convolutional layer contains a lot of feature graphs with different kernels (filters), so that the various lesion characteristics can be extracted at the same sub-region in the image. Its weights are composed of a set of learnable kernels. In order to avoid overfitting, pooling operator is used to reduce the dimensionality of the extracted features on each receptive filed; the fully-connected layer is used to reduce the loss of the image feature information by performing the dot product between the extracted features and the weights. Three extracted feature vectors coming from multiple feature maps in three CNN are integrated as a vector with fully-connected fusion layer. They are again simply concatenated and constituted a high dimensional feature vector to produce the output for classifier. The Softmax activation function is used in the last layer such that each neuron's output activation can be interpreted as the probability of a particular input image belonging to that class. The dropout normalization strategy is used to also avoid the overfitting problem. ReLU is applied as an activation function to the output of every convolutional layer and fully-connected layer to increase the nonlinear properties of the decision function and obtain the robust features. Stochastic gradient descent (SGD) is used to learn the best set of weights and biases of CNN that minimize the loss function. All parameters of TCCNN are optimized at the same time.

3.2. Discriminant information

In order to ensure the proposed TCCNN to achieve a high recognition rate on a small image database, we introduce the discriminant information to the network back propagation to reduce the intra-class distances of the input images and increase the between-class distances of the input images. The overall cost function is defined based on the distance between the intra-class and inter-class scatter matrices:

$$S = R + \gamma S_1 - \beta S_2 \tag{9}$$

where R is the cost function of the TCCNN, γ, β are two adjustment parameters, S_1 and S_2 are inter-class and intra-class scatter matrices, defined respectively as follows,

$$S_1 = \frac{1}{2}\sum_{c=1}^{C}\sum_{i=1}^{n_c}\|x_i^c - M^{(i)}\|^2,$$
$$S_2 = \frac{1}{2}\sum_{i=1}^{C}\sum_{j=i+1}^{C}\|M^{(i)} - M^{(j)}\|^2 \tag{10}$$

where x_i^c is the ith output of the fully-connected fusion belonging to cth class, n_c is the sample number in the cth class, C is the number of the classes, $M^{(i)} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_i$ is the sample average value of the ith class.

The propagation of the network error through the hidden layers can be calculated by the gradient of the overall cost function. When the network weights are adjusted, after calculating the gradients of R, S_1 and S_2, the whole weights are updated by iterating CNN. In conducting the gradient descent algorithm, in every iteration, S_1, as a cost function, is used to enlarge the distance between the

inter-class predicted values, while S_2, as a cost function, is used to decrease the distance between the averages of the intra-class predicted values. By adjusting the weights, TCCNN can achieve quickly the classification purpose when the number of samples is small or the iteration number is small. In this paper, we set the parameters $\gamma = 0.02,\ \beta = 0.01$ by experiments.

3.3. Model training

The proposed method takes RGB diseased leaf images of size of 128×128 as input and the output is a probability vector for each image with one entry for each disease class. The final error is obtained from the output of the three channels. As the classical CNN, TCCNN is also a multi-layer feed-forward neural network to learn features from the input images. It can be trained only by using a known pattern without an accurate mathematical model between any input and output. The trained TCCNN has a mapping ability to learn the feature mapping relationship between the input data and output data of the convolutional layer and pooling layer. Training TCCNN includes two stages: single CNN training and whole model training. In single CNN training, the weights and biases of all channels can be randomly initialized. The weights and biases of some hidden layer nodes in each CNN are randomly changed during the model training. The default of the initial weights for the convolutional layers follows a Gaussian distribution with mean 0 and standard deviation 0.01. The default for the initial bias offset is 0. The whole model weights and biases are fine-tuned by minimizing the categorical cross entropy. The cross entropy represents the dissimilarity of the approximated output distribution (after Softmax) from the true distribution of labels. The training ends when the model does not significantly improve its performance on the validation set for a predefined number of epochs.

3.4. Disease recognition

Two fully-connected layers and a fully-connected fusion layer are used in the recognition stage, where each neuron provides a full connection to all learned feature maps issued from the previous layer in each CNN. These connected layers are based on the Softmax activation function to compute the class scores. The input of the Softmax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class. The Softmax function takes as input a C-dimensional vector z and outputs a C-dimensional vector y of real values between 0 and 1, where C is the number of the disease classes. This function is calculated by Eq. (6).

Each experiment is repeated several times with different training sets to reduce the influence of random effects, and the average results are reported. In the experiments, the overall recognition rate is calculated to infer the robustness of the proposed method. It is the sum of the correctly classified samples divided by the total number of test samples, which is defined as:

$$\text{Recognition rate} = \frac{TrueN}{TotalN} \tag{11}$$

where $TrueN$ is the number of correctly classified samples, $TotalN$ is total number of test samples.

4. Experimental results

To verify the performance of the TCCNN based vegetable disease identification method, we conduct a set of experiments on cucumber and tomato diseased leaf image databases, and compare with four existing vegetable leaf disease recognition methods based on: image processing technology (IPT) (Pixia & Xiangdong, 2013), global-local singular value decomposition (GLSVD) (Zhang et al., 2017), SVM classification algorithm (SVM) (Zhou, Xu, & Zhao, 2015), and sparse representation based classification (SRC) (Guo et al., 2007). In the proposed method, we use color diseased leaf images to directly identify vegetable diseases, instead of segmenting lesion and extracting features. In four comparative methods, several pre-processing process are performed, a lot of features are extracted from each segmented lesion image, and SVM classifier is employed to recognize the diseases. The proposed method and the comparative methods are implemented by MATLAB R2017a, on Intel Core i7-4790 CPU @ 3.60 GHz × 8, GeForce GTX TITAN X 12 GB and 16 GB memory, and Win.7 operating system. Deep-Learn Toolbox (https://github.com/rasmusbergpalm/DeepLearnToolbox) and TensorFlow Toolbox (http://blog.csdn.net/chengyuqiang/article/details/78238001) is used to construct the TCCNN.

4.1. Experiment setup

Image cropping can minimize the foreground portion and then reduce the amount of computation. Fig. 4(A) shows the original input image used for learning, Fig. 4(B) shows the cropped image, and Fig. 4(C) shows the image obtained by resizing the cropped image to the fixed-size of 128×128.

The adjusted images are divided into three components used as the input of the TCCNN. In CNN in Fig. 3(C), the sizes of convolutional kernels are selected as 9×9, 5×5 and 3×3 with a stride of 2, while the size of pooling kernel is 2×2 with non overlapping region. All receptive fields, kernels and feature maps are square in shape. Table 1 lists the different layer sizes and output sizes produced by each layer, where C is the number of disease classes.

As shown in Table 1, the stochastic gradient descent (SGD) algorithm is used in CNN model to learn the best set of weights and biases that minimize the loss function. While learning, SGD works by randomly choosing a small number of training inputs. The mini batch size is set to 10.

(A) Original image (B) cropping image (C) 128×128 image

Fig. 4. Vegetable diseased leaf image cropping and resize example.

Table 1
Different output sizes produced by each layer.

Layer No.	Type	Kernel size	Neuron size	Maps
0	Color image component		128 × 128	
1	Convolutional layer C1	9 × 9	120 × 120	16
2	Pooling layer P1	2 × 2	60 × 60	16
3	Convolutional layer C2	9 × 9	52 × 52	32
4	Pooling layer P2	2 × 2	26 × 26	32
5	Convolutional layer C3	5 × 5	22 × 22	64
6	Pooling layer P3	2 × 2	11 × 11	64
7	Convolutional layer C4	3 × 3	9 × 9	128
8	Fully-connect layer FC1	1 × 1	1 × 1024	1
9	Fully-connect layer FC2	1 × 1	1 × 512	1
10	Softmax	Classifier	1 × 1 × C	C

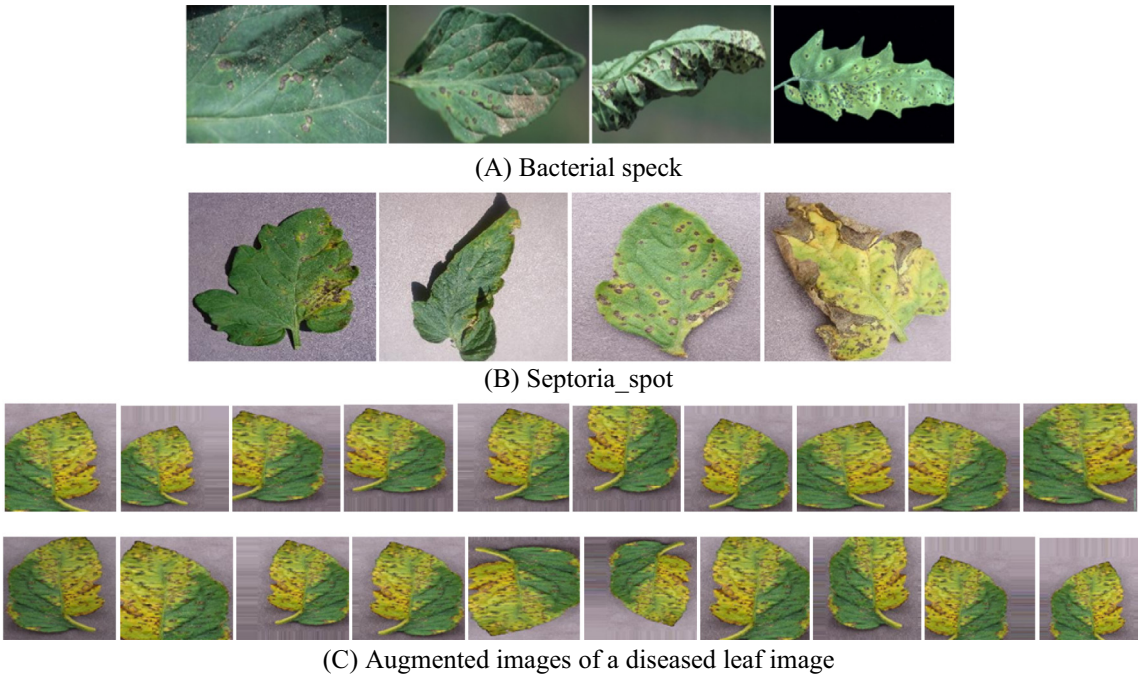
A large learning rate can adjust the weight in the training process, so as to accelerate the speed of network training but it will lead a suboptimal result; while a small learning rate leads to more precise results but it will require more training time. The learning rate is initially set to 0.001 for all layers, and is set to 0.01 for the second fully-connected layer and the fully-connected fusion layer, which is higher than other layers due to the weights being trained starting from random. The learning rate is then decreased by a factor of 10 when the loss decreases slightly. The momentum is an additional factor to determine how fast SGD converges on the optimum point, which is set to 0.9. Every original weight is set as a random number in range of (−0.5, 0.5) of normal distribution with mean of 0 and variance of 1. Every original bias is set 0.

After three channels, the extracted features through three CNN are fused in the fully-connected fusion layer. However, the number of parameters is three times larger than that of a CNN. To implement the TCCNN model and obtain better performance, the weights of the model are initialized from the weights of each CNN. That is to say, each CNN are trained individually by its corresponding color component when the input of the other two CNNs is assigned to zero. The weights of the model can be fine-tuned well (Zhang & Zhang, 2017). It is clearly better than randomly initialized weights.

4.2. Experimental results on Plantvillage tomato leaf database

To further asses the performance of the proposed method, a lot of extend experiments are conducted on the Plantvillage tomato diseased leaf image dataset (https://www.plantvillage.org/en/plant_images). The dataset contains 15,817 color leaf images from 8 kinds of diseases: 2120 bacterial_speck images, 2572 early_blight images, 1903 late_blight images, 945 leaf_mold images, 1164 septoria_leaf_spot images, 1397 target_spot images, 366 mosaic_virus images and 5350 yellow_leaf_curl_virus images. It is noted that each image was taken with simple background, and several images of the same leaf were taken from different orientations. All original diseased leaf images are augmented 20 times (Leng, Yu, & Qin, 2017). Fig. 5 shows some tomato diseased leaf examples and 20 augmented images by a diseased leaf image.

In the paper, the original database is expanded by randomly rotating, skewing, interpolating, mirroring, adjusting contrast, and introducing slight distortion to the original images, and adding a noise (Gaussian with 3 pixels) in the original images to consider a kind of controlled environment. In the training process of each image, the displacement, rotation and scale transform are performed by using some random bounded values. These values are



(A) Bacterial speck

(B) Septoria_spot

(C) Augmented images of a diseased leaf image

Fig. 5. Tomato diseased leaf examples.

evenly distributed in a specific range, and the image displacement is in the range of ±10%, the scale is changed from 0.9 to 1.1, and the rotation range is ±5°. We also increase the training image set by altering the intensities of the RGB channels in training images. For each image, 20 additional synthetic images were generated, as shown in Fig. 5C, and then each image is cropped to the fixed-size of 128 × 128 pixels to fit the model. Ten-fold-cross validation scheme is adopted to split the dataset into training and testing subsets. That is to say, all of the images per class are randomly split into 10 almost equally sized subsets. In turn, each of the 10 subsets is used once as test subset for testing the performance of the proposed method, while the remaining nine subsets are used as the training set to train the TCCNN and classifier. We divide each tomato RGB disease image into three components R, G and B, each component with same size of 128 × 128. The components are normalized from [0, 255] to [−1, 1]. Finally, the performance measures of all ten splits are averaged. The above experimental process is repeated 50 times and 50 average results are averaged again as the final recognition rate. Table 2 shows the disease recognition results on tomato diseased leaf image database.

To further test the performance of the TCCNN, we compare with DNN (29), LeNet-5 (Glauner, 2015) and GoogleNet (Dyrmann, Karstoft, & Midtiby, 2016). As the above experimental steps, a lot of experiments are conducted. Ten-fold-cross validation scheme is adopted, and every split is repeated 50 times and the results are averaged. The disease recognition results are shown in Table 3.

4.3. Experimental results on cucumber diseased leaf image database

The proposed method is firstly tested on our cucumber diseased leaf image database. All diseased leaf images of the database were collected by Android mobile phone, Canon Camera and Internet of things in the agricultural demonstration zone of Northwest Agriculture and Forestry University in China. In the experiments, 500 cucumber diseased leaf images with 100 images per class are selected from five kinds of cucumber diseases: Scab Angular, Powdery Mildew, Downy Mildew, Anthracnose and Scab. Some diseased leaf images are shown in Fig. 6. These images are captured with different sizes, orientation, locations, illumination and backgrounds.

All original diseased leaf images are augmented 14 times and then divided into a training set and a test set with different percentage of the images to validate the effectiveness of the proposed method. To know how the proposed model will perform on new unseen data, and to keep a track of if any of the proposed method is overfitting, we run all our

Table 2
The disease recognition rates results (%) and variants by IPT, GLSVD, SVM, SRC and the proposed method on the tomato disease image database.

Method	IPT	GLSVD	SVM	SRC	Our method
Recognition result	58.24 ± 3.62	62.28 ± 3.13	61.37 ± 2.64	60.25 ± 3.52	87.15 ± 2.17

Table 3
The disease recognition rates results (%) and variants by DNN, LeNet-5, GoogleNet and the proposed method on the tomato disease image database.

Method	DNN	LeNet-5	GoogleNet	Our method
Recognition result	84.37 ± 2.56	89.83 ± 2.61	90.13 ± 2.24	91.15 ± 2.17

(A) Scab Angular (B) Powdery Mildew (C) Downy Mildew (D) Anthracnose (E) Scab

Fig. 6. Five kinds of cucumber diseased leaf images.

Table 4
Disease recognition rates (%) and variants by IPT, GLSVD, SVM, SRC and the proposed method on the original cucumber disease image database.

	Method				
Train/Test	IPT	GLSVD	SVM	SRC	Our method
80/20	51.62 ± 2.73	56.42 ± 3.41	57.24 ± 2.17	67.52 ± 2.15	91.16 ± 2.16
70/30	47.23 ± 3.28	47.37 ± 3.32	50.80 ± 3.24	64.57 ± 2.36	89.29 ± 2.34
60/40	44.62 ± 3.24	42.58 ± 3.64	47.74 ± 3.13	62.14 ± 3.18	88.13 ± 2.45
50/50	42.76 ± 3.61	42.42 ± 3.58	41.68 ± 2.35	58.87 ± 2.74	87.57 ± 2.61
40/60	41.63 ± 3.53	40.37 ± 3.73	40.33 ± 2.62	57.43 ± 3.12	85.32 ± 2.76

Table 5
Disease recognition rates (%) and variants by IPT, GLSVD, SVM, SRC and the proposed method on the cucumber segmented lesion image database.

	Method				
Train/Test	IPT	GLSVD	SVM	SRC	Our method
80/20	91.32 ± 2.95	92.17 ± 1.84	91.51 ± 2.49	92.86 ± 2.14	94.15 ± 2.48
70/30	88.53 ± 2.26	90.52 ± 2.52	89.66 ± 2.61	91.39 ± 2.35	94.27 ± 2.51
60/40	85.12 ± 2.31	87.36 ± 2.81	86.93 ± 3.45	89.41 ± 2.46	93.26 ± 2.73
50/50	82.24 ± 3.14	82.42 ± 2.36	84.83 ± 3.53	86.69 ± 2.91	91.64 ± 2.84
40/60	79.15 ± 3.35	81.29 ± 2.95	81.57 ± 3.82	79.15 ± 3.44	90.15 ± 2.98

experiments across a whole range of train/test set splits, namely m_1/m_2 (m_1% of the whole dataset used for training, and m_2% for testing). For each dataset split, we train the TCCNN and repeat the recognition experiment 50 times. That is, each experiment is repeated 50 times with different training sets to reduce the influence of random effects, and the average results are reported. Table 4 shows the disease recognition results of the proposed method and four comparative methods on the disease plant image database. For comparison, Table 5 also shows the disease recognition results of these methods on the corresponding segmented lesion image database.

From Tables 2, 4 and 5, it is seen that the proposed method outperforms the state-of-the-art methods which used the hand-crafted features, even if the optimal classifier is adopted. The main reason is that the proposed approach makes use of the color characteristic of the diseased leaf image, and can automatically learn more high-level discriminant features than the four feature extraction based plant disease identification methods, and the learnt higher-level features from the diseased leaf image can represent the particular characteristics of diseases. While the performances of the four compassion approaches depend heavily on image preprocessing, image enhancement, lesion segmentation and hand-engineered features, and it is difficult to decide the most distinctive subset of features for disease recognition. From Table 3, it is found that the proposed method is better than other one DNN and two CNN based diseased leaf image classification methods. The reason may be the proposed method makes full use of the color characteristic of the diseased leaf image, and employs the class discriminant information to the adjustment process of the network back propagation weight, which is beneficial to classification.

5. Conclusion

In this paper, a vegetable disease recognition approach is presented based on TCCNN. In the method, the high-level discriminant features can be automatically extracted through TCCNN directly from the color diseased leaf image, instead of complex preprocessing, lesion segmentation and hand-crafted feature extraction. The proposed method avoids the lesion segmentation and hand-crafted feature process. The experimental results demonstrate the multi-channel CNN is effective and feasible.

Conflict of interest statement

The authors declare that there is no potential conflict of interest referring to this article.

Acknowledgement

This work is supported by the China National Natural Science Foundation under grant Nos. 61473237 & 61472280, Key Research and Development Projects (2017ZDXM-NY-088), Key Project (2016GY-141) of Shaanxi Department of Science and Technology and Tianjin Jinnan Technology Research Program (2017). The authors would like to thank all the editors and anonymous reviewers for their constructive advice.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.cogsys.2018.04.006.

References

Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. *Lecture Notes in Informatics*, 79–88.

Bao, W., Chen, Y., & Wang, D. (2014). Prediction of protein structure classes with flexible neural tree. *Bio-Medical Materials and Engineering, 24*(6), 3797–3806.

Bao, W., Wang, D., Chen, Y., et al. (2017). Classification of protein structure classes on flexible neutral tree. *IEE ACM Transactions on Computational Biology and Bioinformatics, 14*(5), 1–12.

Barbedo, A. (2016). A review on the main challenges in automatic plant disease identification based on visible range images. *Biosystems Engineering, 144*, 52–60.

Dyrmann, M., Karstoft, H., & Midtiby, H. S. (2016). Plant species classification using deep convolutional neural network. *Biosystems Engineering, 151*, 72–80.

Glauner, P. O. (2015). Deep convolutional neural networks for smile recognition. *IEEE ACM Transactions on Audio Speech & Language Processing, 22*(10), 1533–1545.

Guo, Y., Hastie, T., & Tibshirani, R. (2007). Regularized linear discriminant analysis and its application in microarrays. *Biostatistics, 8*(1), 86–100.

He, A. & Tian, X. (2016). Multi-organ plant identification with multi-column deep convolutional neural networks. In *IEEE international conference on systems, man and cybernetics* (pp. 2020–2025). Budapest.

Huang, D. S. (1996). *Systematic theory of neural networks for pattern recognition*. Publishing House of Electronic Industry of China.

Huang, D. S. (1999). Radial basis probabilistic neural networks: Model and application. *International Journal of Pattern Recognition & Artificial Intelligence, 13*(07), 1083–1101.

Huang, D. S. (2004). A constructive approach for finding arbitrary roots of polynomials by neural networks. *IEEE Transactions on Neural Networks, 15*(2), 477–491.

Huang, D. S., Jiang, W. (2012). A general CPL-AdS methodology for fixing dynamic parameters in dual environments. *IEEE Transactions on Systems, Man and Cybernetics-Part B, 42*(5), 1489–1500.

Huang, D. S., & Du, J. X. (2008). A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks. *IEEE Transactions on Neural Networks, 19*(12), 2099–2115.

Huang, D. S., Ip, H. H. S., Law, K. C. K., et al. (2005). Zeroing polynomials using modified constrained neural network approach. *IEEE Transactions in Neural Networks, 16*(3), 721–732.

Huang, D. S., Huang, Z., Yuan, C. A., et al. (2017). Pupylation sites prediction with ensemble classification model. *International Journal of Data Mining & Bioinformatics, 18*(2), 91–104.

Huang, D. S., Ip, H. H. S., & Chi, Z. R. (2004). A neural root finder of polynomials based on root moments. *Neural Computation, 16*(8), 1721–1762.

Huang, W., & Zhang, S. (2017). A novel face recognition algorithm based on the deep convolution neural network and key points detection jointed local binary pattern methodology. *Journal of Electrical Engineering & Technology, 12*(1), 363–372.

Leng, B., Yu, K., & Qin, J. (2017). Data augmentation for unbalanced face recognition training sets. *Neurocomputing, 235*, 10–14.

Li, Bo, & Huang, D. S. (2008). Locally linear discriminant embedding: An efficient method for face recognition. *Pattern Recognition, 41*(12), 3813–3821.

Li, S., Huang, D. S., Du, J.-X., et al. (2006). Palmprint recognition using Fast ICA algorithm and radial basis probabilistic neural network. *Neurocomputing, 69*(13–15), 1782–1786.

Liu, K.-H., & Huang, D. S. (2008). Cancer classification using rotation forest. *Computers in Biology and Medicine, 38*(5), 601–610.

Mohanty, S. P., Hughes, D. P., & Marcel, S. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*. https://doi.org/10.3389/fpls.2016.01419.

Pixia, D., & Xiangdong, W. (2013). Recognition of greenhouse cucumber disease based on image processing technology. *Open Journal of Applied Sciences, 3*, 27–31.

Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *IEEE Conference on Computer Vision & Pattern Recognition*, 3642–3649.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85–117.

Srdjan, S., Marko, A., Andras, A., et al. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational Intelligence and Neuroscience, 6*, 1–11.

Wang, X. F., & Huang, D. S. (2009). A novel density-based clustering framework by using level set method. *IEEE Transaction on Knowledge and Data Engineering, 21*(11), 1515–1531.

Wang, X. F., Huang, D. S., & Xu, H. (2010). An efficient local Chan-Vese model for image segmentation. *Pattern Recognition, 43*(3), 603–618.

Wiatowski, T., & Bölcskei, H. (2016). A mathematical theory of deep convolutional neural networks for feature extraction. *Mathematics*, 1–50.

Wright, J., Yang, A. Y., Ganesh, A., et al. (2009). Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence, 31*(2), 210.

Zhang, S. & Zhang, C. (2017). Plant species recognition based on deep convolutional neural networks. In *International conference on intelligent computing* (pp. 282–289). Springer, Cham.

Zhang, S., & Wang, Z. (2016). Cucumber disease recognition based on Global-Local Singular value decomposition. *Neurocomputing, 205*, 341–348.

Zhang, S., Wu, X., You, Z., et al. (2017). Leaf image based cucumber disease recognition using sparse representation classification. *Computers & Electronics in Agriculture, 134*, 135–141.

Zhao, W. B., Huang, D. S., & Guo, L. (2003). Comparative study between radial basis probabilistic neural networks and radial basis function neural networks. *Lecture Notes in Computer Science, 2690*, 389–396.

Zhao, Z. Q., Huang, D. S., & Sun, B. Y. (2004). Human face recognition based on multi-features using neural networks committee. *Pattern Recognition Letters, 25*(12), 1351–1358.

Zhou, B., Xu, J. & Zhao, J. (2015). Research on cucumber downy mildew detection system based on SVM classification algorithm. In *3rd International conference on material, mechanical and manufacturing engineering (IC3ME)* (pp. 1681–1684).