Research on crop disease recognition based on uniting multi-layer features

Peng Ni¹, Zhen Chen² and Maoyong Cao¹*

¹ College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, Shandong, 266000, PRC
² College of Electrical Engineering and Automation, Qilu University of Technology, Jinan, Shandong, 250000, PRC
*Corresponding author’s e-mail: cmy@qlu.edu.cn

Abstract. Traditional image processing has many problems in crop disease recognition, such as complicated manual design and low efficiency. This article studies the application of deep learning algorithms in crop disease recognition. This article analyzes the characteristics of feature maps extracted from different convolutional groups of ResNet, proposes multi-layer features united ResNet (MLFU-ResNet) for crop disease classification, and introduces label smoothing to modify the loss function. The experimental results show that the network structure proposed in this article has achieved better classification performance in crop disease data sets. Compared with ResNet, the classification accuracy of the network structure proposed in this article has increased by 0.8%.

1. Introduction
The occurrence of crop diseases is one of the important disasters that cause crop yields to decrease. The timely detection and diagnosis of crop diseases is the basis and key to comprehensive disease prevention. In addition, improper application of pesticides in the process of disease prevention and control will also cause environmental pollution and food pollution. Therefore, the accurate recognition of crop diseases and the recommendation of appropriate prevention and control measures can not only ensure the yield of crops but also reduce various hazards caused by the use of pesticides[1].

Traditional crop disease recognition is carried out by extracting and screening the color, texture, shape, and other characteristics of disease images. However, the pathological characteristics of the same disease at different disease stages are very small, and multiple diseases may show similar pathological characteristics. These complex natural factors make the traditional recognition methods less universal in solving the problem of crop disease recognition.

In recent years, with the development of deep learning technology, more and more researchers use deep learning technology to solve the problem of crop disease recognition and have made great progress[2]. In 2016, Mohanty et al.[3] used GoogLeNet to classify 14 crops and 26 types of diseases and achieved a classification accuracy of more than 90% on the test set. In 2017, Pennsylvania State University Ramcharan et al.[4] used transfer learning to train a deep neural network model in the classification of cassava pests and diseases, and the classification accuracy of the five cassava pests and diseases reached more than 95%. Zhang Shanwen et al.[5] designed an 11-layer LeNet to train the collected 1200 color pictures of cucumber diseases. Experiments show that the classification accuracy of this method has reached more than 90%. It can be seen from this that deep learning methods can
quickly learn the features required for classification tasks from large-scale data sets, and can achieve better efficiency and accuracy in crop disease recognition tasks. However, most of the existing researches only identify the types of crop diseases, and few assess the severity of crop diseases. In this article, ResNet is optimized from two aspects: model structure and loss function. Good classification accuracy was achieved in the task of assessing the severity of crop diseases.

2. Structural design of MLFU-ResNet

This study is based on the deep learning method to classify and recognize crop disease images. First, load the data sets, perform online preprocessing on the data sets, and then send the data sets to the built-up MLFU-ResNet for training, and perform feature extraction and multi-layer features united. Finally, the acquired features are evaluated for the severity of crop diseases through the SoftMax function. The overall algorithm framework is shown in figure 1. Among them, $P_n$ represents the probability that it is predicted to be the nth disease.

![Algorithm flowchart](image)

Figure 1. Algorithm flowchart.

2.1. ResNet-50 model

ResNet[6] was proposed in 2016. Because the network "simple and practical" coexist, many subsequent network models are completed basic on this network and have been widely used in object detection, segmentation, recognition, and other fields. The network is implemented by stacking residual units. This structure allows the network model to be deepened while still maintaining high feature expression capabilities and solves the problem of gradient degradation caused by the excessively deep network structure. The structure of the residual unit is shown in figure 2.
Weight Layer

\[ F(X,W_i) \]

relu

\[ H(X) = F(X,W_i) + X \]

Different from the traditional connection method, the residual unit adopts a cross-layer connection method to directly bypass the input information to the output, creating an identical mapping of the input, and protecting the integrity of the information. Without increasing the weight parameters and the difficulty of calculation, effectively suppress the vanishing gradient. The output of each layer of the residual unit can be expressed as:

\[ H(X) = F(X,W_i) + X \]
\[ F = W_2 \sigma(W_1X) \]

Among them, \( X \) is the input information; \( W \) is the weight parameter that needs to be learned; \( \sigma(\cdot) \) is the activation function; \( F(\cdot) \) is the calculation result of the convolutional layer on \( X \).

In addition, the residual connection also breaks the symmetry of the network[7], solves the network degradation problem caused by the deep neural network, and improves the characteristic representation ability of the network. The specific network parameters of ResNet-50 is shown in the longitudinal part of table 1, where \([\cdot]\) represents a group of residual unite.

2.2. MLFU-ResNet structure design

In order to have a better understanding of the features extracted by each convolution group in ResNet, this article uses visualization methods to visualize the feature maps extracted by different convolution groups. Figure 3 shows the visualization effect of the feature maps of the partial convolution group of ResNet-50.

Analysis of figure 3 shows that the features extracted by the deep convolution group that closer to the output are more abstract and representative, in addition, the spatial resolution is lower. The
features extracted by the shallow convolution group that far from the output are specific features such as contours and shapes, but they also have some key features, such as the highlighted part in figure 3(d)[8]. In ResNet-50, only the features extracted from the last feature group are used as final feature representations, and specific features such as contours and shapes extracted from the shallow convolution group cannot be fully utilized.

As shown in figure 4, to make full use of the different features extracted by the shallow and deep convolutional groups of ResNet. Based on ResNet-50, this article additionally selects the feature output of conv1, conv2, and conv3 as nodes, adding three horizontal connections. First, based on the output feature maps of conv1, conv2, and conv3, through convolution and pooling operations, feature extraction and down-sampling are performed again. In this process, only the size of the feature map is changed, and the dimension of the feature map is not changed. After that, the three groups of downsampled features and the output features of conv4 are deeply united by concatenating to form a composite feature based on the features of the deep convolution group and supplemented by the features of the shallow convolution group. Then the composite features are passed through the refine block, which has two effects: (1) Channel compression, compressing 3840 channels to 2048 channels; (2) enabling the four groups of features to be better united. The specific structure and parameters of MLFU-ResNet are shown in figure 4 and table 1.

![Figure 4. MLFU-ResNet structure.](image)

| Layer name (Longitudinal part) | Parameter | Layer name | Parameter |
|-------------------------------|-----------|------------|-----------|
| Stem                          | Conv 7 × 7, 64, S=2 | Refine1    | Conv       |
|                               | Maxpool 3 × 3, S=2   |            | 1 × 1, 256, S=1, P=0 |
|                               |                        |            | 3 × 3, 256, S=2, P=1 |
|                               | Refine2               |            | Avgpool    |
|                               | 1 × 1,128, S=2        |            | 2 × 2, S=2 |
|                               | 3 × 3,128, S=2        |            |            |
|                               | 1 × 1,512             |            |            |
|                               | x 4                   |            |            |
|                               | Refine3               |            | Conv       |
|                               | 1 × 1,256, S=2        |            | 1 × 1, 256, S=1, P=0 |
|                               | 3 × 3,256, S=2        |            | 3 × 3, 512, S=2, P=1 |
|                               | 1 × 1,1024, S=1, P=0  |            |            |
|                               | x 4                   |            |            |
|                               | Cat                   |            |            |
|                               |                      |            | 3 × 3, 2048, S=2, P=1 |
|                               |                      |            |            |
|                               |                      |            |            |
|                               | 1 × 1,512             |            |            |
|                               | 3 × 3,512, S=2        |            |            |
|                               | 1 × 1,2048            |            |            |
|                               | x 3                   |            |            |
|                               | Refine                |            |            |
|                               | Global Avgpool        |            |            |
|                               | 59, FC                |            |            |
|                               | softmax               |            |            |
|                               |                      |            |            |
|                               | Global Avgpool        |            |            |
|                               | 59, FC                |            |            |
|                               | softmax               |            |            |
3. Experiment and result analysis

3.1. Data sets introduction and experimental environment
The data set used in this article is from AI Challenger 2018 (https://challenger.ai/competition/pdr2018), classified according to species-disease-degree, including 10 species, 27 diseases, and divided into 61 according to the degree of disease incidence class. The labeled data includes 31718 training images and 4540 validation images. Figure 5 shows some crop disease images.

![Crop disease images](a. Apple healthy b. Apple scab general c. Apple scab serious d. Pepper healthy e. Pepper scab general f. Pepper scab serious)

Figure 5. Crop disease images.

Through the analysis of the data sets, We decide to delete the two categories 44 and 45. Before use, cleaning the data set by deleting pictures with cross-labels. The processed data set includes 59 categories, 31493 training images, and 4527 verification images.

In the experiment, the following methods are used for online data preprocessing on the data set: random horizontal flip, random vertical flip, and random angle flip. These data preprocessing methods do not change the original color of the image and are mainly used to simulate the randomness of shooting angles under natural conditions. And the image RGB three-channel pixels are normalized to [-1,+1] so that the distribution of all images is similar. These pre-processing methods make the training set richer and the model has more generalization capabilities, and the normalization operation makes the model easier to converge during training, reducing model training time.

The experiments in this paper are all based on the deep learning framework PyTorch. The computer environment is a Linux system, the CPU model is Intel(R) Xeon Gold 5118 cpu @ 2.30GHz; the GPU model is GeForce RTX 2080 Ti 11GB.

3.2. Model parameter setting
In the experiment, the cross-entropy loss function[9] is used to train the optimal model, and the label smoothing[10] is introduced to modify the cross-entropy loss function. In a multi-classification task, the neural network model will output a confidence score corresponding to each category of the current data, and normalize these scores through softmax, and finally get the probability $q_i$ that the current data belongs to each category.

$$q_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$ (3)

Then calculate the cross-entropy loss function:

$$Loss = - \sum_{i=1}^{K} p_i \log q_i$$ (4)

Where $p_i$ is the true probability distribution.

$$p_i = \begin{cases} 1, & (i = y) \\ 0, & (i \neq y) \end{cases}$$ (5)
When training a neural network, it is necessary to use the predicted probability to fit the true probability, but fitting the one-hot true probability function will bring two problems:

1) The generalization ability of the model cannot be guaranteed;
2) Full probability and 0 probability encourage the gap between the category and other categories to be as wide as possible. This causes the model to trust the predicted category too much.

When the training data is small and insufficient to characterize all the sample features, it will cause the network to overfit.

Label smoothing uses a soft one-hot to add noise, which reduces the weight of the real sample label category in the calculation of the loss function, and finally achieves the effect of suppressing overfitting.

After adding label smoothing, the true probability distribution will become:

\[
p_i = \begin{cases} 
(1 - \varepsilon), \text{if } (i = y) \\
\varepsilon/K - 1, \text{if } (i \neq y)
\end{cases}
\]  

(6)

Where:
\(K\) — the total number of categories in multiple categories
\(\varepsilon\) — label smoothing adjustment factor

The cross-entropy loss function will eventually become the following form:

\[
Loss_i = \begin{cases} 
(1 - \varepsilon) \cdot Loss, \text{if } (i = y) \\
\varepsilon \cdot Loss, \text{if } (i \neq y)
\end{cases}
\]  

(7)

In terms of parameter settings, choose the Adam optimizer that is simple to implement and computationally efficient. The input image size is 224x224, the initial learning rate is set to 0.0001, the learning rate adjustment method is decline 10 times every 15 epochs, the Batch size is 32, the Dropout is set to 0.7, and the number of epochs is set to 100.

3.3. Experimental results

This article uses accuracy to evaluate the performance of the model. The accuracy index is defined as follows:

\[
Acc = \frac{N_T}{N} \times 100\%
\]  

(8)

where:
\(N_T\) — the number of validation samples that are predicted to be correct
\(N\) — the total number of validation samples

To verify the classification performance of MLFU-ResNet, we choose to train ResNet and MLFU-ResNet separately using the cross-entropy loss function and the cross-entropy loss function modified by label smoothing in the same environment. Table 2 shows the classification accuracy of different models and different loss function training. The results show that the best classification accuracy is achieved in the crop disease data set when the MLFU-ResNet is trained with the cross-loss function modified by the label smoothing, and the accuracy rate reached 87.89%, compared with ResNet trained with the cross-entropy loss function, the accuracy is increased by 0.8%.

Table 2. Experimental result.

| Model                              | Acc   |
|------------------------------------|-------|
| ResNet                             | 87.09%|
| ResNet+LabelSmoothing(\(\varepsilon\)=0.14) | 87.12%|
| MLFU-ResNet                        | 87.78%|
| MLFU-ResNet+LabelSmoothing(\(\varepsilon\)=0.14) | 87.89%|

* label smoothing adjustment factor
4. Conclusion
This article visualizes the features extracted from different convolution groups of ResNet, analyzes the characteristics of the features extracted from different convolution groups and the shortcomings of using the features extracted from the last group of convolution groups as the final feature representation. Based on this, this paper proposes the MLFU-ResNet by uniting multi-layer features. This network model makes full use of the specific features such as contours and shapes extracted by the shallow convolution group and the abstract and representative features extracted by the deep convolution group. In addition, label smoothing technology is introduced to modify the cross loss function to enhance the generalization ability of the network. The final experimental results show that compared to ResNet, the classification accuracy of MLFU-ResNet on the crop disease data sets has been improved by 0.8%, and it has achieved better classification performance.

References
[1] Kang Fj, et al. (2020) Application technology of image recognition for various crop diseases and insect pests: a review[J]. Jiangsu Agricultural Sciences, 48: 22-27.
[2] Jia Sp, et al. (2019) Research Progress on Image Recognition Technology of Crop Pests and Diseases Based on Deep Learning[J]. Transactions of the Chinese Society for Agricultural Machinery, 50:313-317.
[3] Mohanty SP, et al. (2016) Using Deep Learning for Image-Based Plant Disease Detection[J]. Frontiers in Plant Science, 7:1419-1425.
[4] Amanda R, Kelsee B, Peter MC, et al. (2017) Deep Learning for Image-Based Cassava Disease Detection[J]. Frontiers in Plant Science, 8: 1852-1859.
[5] Zhang Sw, Xie Zq, et al. (2018) Application research on convolutional neural network for cucumber leaf disease recognition [J]. Jiangsu Journal of Agricultural Sciences, 34: 56-61.
[6] He Km, et al. (2016) Deep Residual Learning for Image Recognition[C]. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Las Vegas. 770-778.
[7] Shang W, Sohn K, Almeida D, et al. (2016) Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units[J]. 2217-2225
[8] Zeiler M D, Fergus R. (2014) Visualizing and understanding convolutional networks[C]. In: Computer Vision ECCV. 2014 [S.L]: Springer International Publishing, Zurich. 818-833.
[9] Boer P, Kroese D P, Mannor S, et al. (2005) A Tutorial on the Cross-Entropy Method[J]. Annals of Operations Research, 134: 19-67.
[10] Szegedy C, Vanhoucke V, Loffe S, et al. (2016) Rethinking the Inception Architecture for Computer Vision. In: IEEE 2016 IEEE conference on computer vision and pattern recognition (CVPR). Las Vegas. 2818-2826.