Hindawi
Journal of Sensors
Volume 2019, Article ID 2716975, 9 pages
https://doi.org/10.1155/2019/2716975

Research Article

Research on Classification Method of Maize Seed Defect Based on Machine Vision

Sheng Huang, Xiaofei Fan, Lei Sun, Yanlu Shen, and Xuesong Suo

College of Mechanical and Electrical Engineering, Hebei Agriculture University, Baoding Hebei Province 071000, China

Correspondence should be addressed to Xuesong Suo; 13903120861@163.com

Sheng Huang and Xiaofei Fan contributed equally to this work.

Received 2 July 2019; Revised 9 September 2019; Accepted 20 September 2019; Published 25 November 2019

Academic Editor: Valerie Renaudin

Traditionally, the classification of seed defects mainly relies on the characteristics of color, shape, and texture. This method requires repeated extraction of a large amount of feature information, which is not efficiently used in detection. In recent years, deep learning has performed well in the field of image recognition. We introduced convolutional neural networks (CNNs) and transfer learning into the quality classification of seeds and compared them with traditional machine learning algorithms. Experiments showed that deep learning algorithm was significantly better than the machine learning algorithm with an accuracy of 95% (GoogLeNet) vs. 79.2% (SURF+SVM). We used three classifiers in GoogLeNet to demonstrate that network accuracy increases as the depth of the network increases. We used the visualization technology to obtain the feature map of each layer of the network in CNNs and used the heat map to represent the probability distribution of the inference results. As an end-to-end network, CNNs can be easily applied for automated seed manufacturing.

1. Introduction

Maize is one of the most important crops global-wise. About one-third of the world's population consumes maize as the major food source. Due to the urbanization, the cultivated land has been decreasing, which is a prominent issue particularly in China. The quality of seeds has become a growing concern for us. The phenotypic defects of seeds are one of the criteria for judging the quality. The traditional method of detecting seed defects typically relies on manual inspection, which is inefficient and subjective. Therefore, an objective and automated seed screening method is required.

Researchers have applied machine vision technology to achieve seed quality testing [1–4]. Features, such as color, texture, size, and shape, can be extracted from images of seeds, and the defects of the seed can be identified through various classifiers based on computer vision. This procedure can be easily automated and thus provide a significantly more efficient method for seed sorting than being inspected by human labor.

In recent years, deep learning has been rapidly developed. For example, some search engines, recommendation systems, image recognition, and speech recognition have adopted deep learning techniques and achieved good results [5]. As the performance of the GPU and the power of parallel computing continues to improve, it is possible to process graphical data in real time. Excellent results have been achieved with convolutional neural networks (CNNs) for image recognition. CNNs have obvious advantages over ordinary machine learning algorithms: the deep convolutional neural network is more independent on the display structure of the data during the image processing and weight sharing, down sampling, and local receptive fields can be used in the network [6]. The data processing method guaranteed the high efficiency of the CNNs. In the process of using CNNs, we only need to determine certain parameters, such as learning rate, weight loss, and batch size, of the optimization algorithm. At the same time, the accuracy of the algorithm increases with the amount of data we obtain. However, CNNs also have their disadvantages: (1) there are thousands

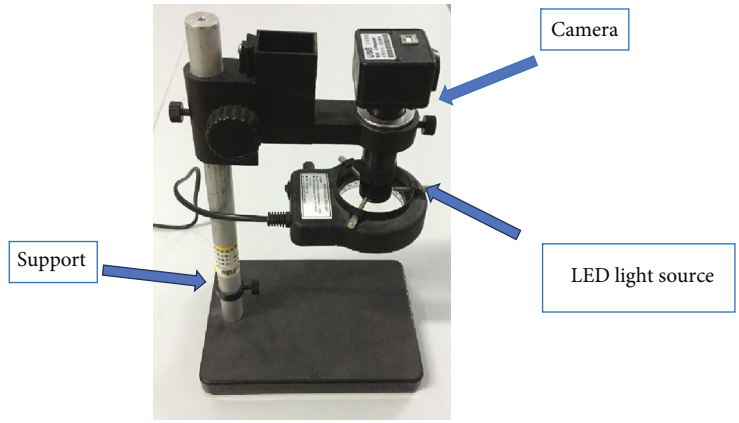


Figure 1: The image acquisition equipment.

of parameters in the model, and it is a long process to tune the parameters. (2) There are a lot of redundant information in the large number of feature maps; we cannot artificially intervene to filter out useful information. Researchers have also applied CNNs to the task of seed identification. Heo et al. used CNNs to filter weed seeds from high-quality seeds [7]. Veeramani used CNNs to distinguish between haploid and polyploid maize seeds [8], Ravindran used transfer learning to classify wood species [9], and Uzal used CNNs to estimate the number of soybean seeds [10].

The purpose of this study was to use CNNs and transfer learning to identify the appearance defects of maize seeds. We used CNNs to compare with traditional machine learning algorithms; the relationship between network depth and the accuracy of CNNs was also studied.

2. Materials and Methods

2.1. Data Collection. We used a "1/2.5" CMOS camera (MindVision MD-U500) with a 28 mm lens (MindVision ML15). The maximum resolution of 2592∗1944 and an FPS of 10. The depth of each color channel is 8 bits, and a white LED ring light source (MindVision MD-HX24) is used. A black background was used to make the maize seeds and background more distinguishable.

The seed image library was acquired by our custom-made image acquisition equipment (Figure 1). In this study, 4000 corn seeds were used and divided into training and testing sets, 20% of the seeds are randomly selected as test sets. Table 1 shows the number of divided data sets. Both the training set and testing set were composed of two groups of corn seeds, respectively. One group of seeds were defect free in appearance, and the other group was with defects including, mold, worm, damages, and discoloration (see Figure 2).

2.2. Image Segmentation. In the actual testing process, it is unrealistic to test only one seed at a time, so we imaged multiple seeds at the same time. The problem was that there might be seeds getting in touch with each other, which greatly affected the classification of single seeds. Many researchers have open-sourced projects for target detection. These frameworks are based on CNN and enable identification of multiple targets in a single image simultaneous. The best performing frameworks are Faster R-CNN [11], SSD [12], and YOLO [13]. Faster R-CNN showed excellent results in the agricultural sector [14]. However, due to the large amount of computation in the training process, high-performance GPU is required to participate in the operation. Therefore, we used image processing algorithms for seed singulation.

Table 1: The number of divided data sets.

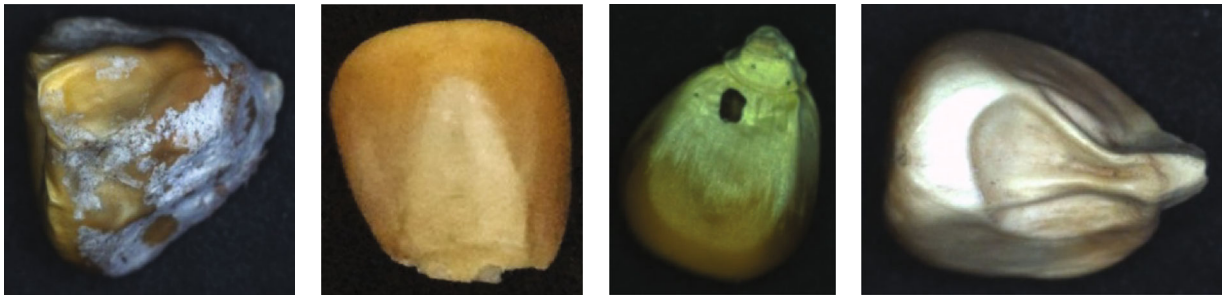
	Training set	Testing set
Worm	376	94
Mold	496	117
Damages	356	89
Discoloration	286	71
Good	1714	428

We used state-of-the-art image processing techniques with OpenCV [15] to segment seeds from the images. Firstly, the original color image was converted into grayscale image, and the maximum interclass variance method (OTSU) [16] was used to obtain the binary image of the approximate contour of the seed. The morphological operations were then used for removing noise in the background and holes in the seeds area.

The next step was to dilate the image so that the image of the real seed was a subset of the expanded image. We used distance transform to find the center area of each seed, and then subtracted the dilated image from the central region to obtain an indeterminate seed edge. Lastly, we used watershed algorithm to get the exact position of each seed edge, and different colors were used to represent each seed for easy observation. The process was shown in Figure 3(a).

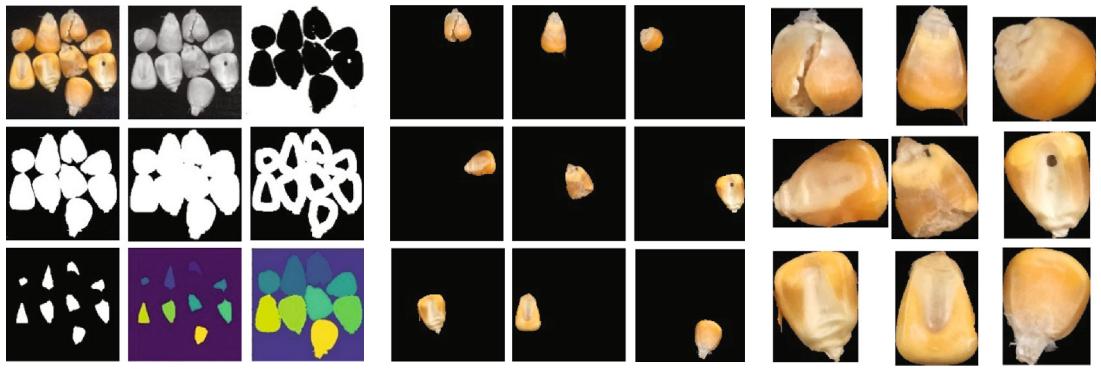
The final step was to extract each seed from the original image. We calculated the position coordinates of each seed. In order to prevent interference with adjacent seed images during the segmentation process, we only kept one maize seed in the original image before cropping (Figure 3(b)); the final results were shown in Figure 3(c).

2.3. Data Augmentation. Since there were limited data in the training set, increasing the number of pictures before starting training would increase the accuracy of CNNs [17], and



(a) (b) (c) (d)

Figure 2: The seeds with defects.



(a) (b) (c)

Figure 3: The process of dividing an image containing seeds connected to each other into a single seed image. (a) The process of finding the edge of the seed using the watershed algorithm. (b) Single seed image before cropping in the original image. (c) The images of seed after cropping.

considering uncertain factors such as the placement angle and position of the seed during the recognition process, we randomly flipped, moved, rotated the images, and applied the transformed images in training along with the original images to improve the robustness of the model. The total number of seeds is 20000.

2.4. Machine Learning Algorithm. A traditional computer vision method requires extracting features, such as shape, color, and texture, from the original image for the training process [3], which is usually overly complicated, so we adopted Speeded Up Robust Features (SURF) algorithm [18] and used a variety of classifiers for comparison in this study.

The location and direction of the seeds could be random during the classification process, and the SURF algorithm was used to extract the features from each seed image. The algorithm has the characteristics of scale and rotation invariance and uses an integral map to calculate the convolution, then use the Hessian response to measure whether a point is a feature point and create a descriptor to describe the feature.

We used logistic regression, support vector machine (SVM), K-nearest neighbor (KNN), and ensemble learning to classify the extracted features: (1) logistic regression is a simple classifier which models the mean response as a function of the linear combination of predictors. (2) Medium Gaussian SVM is a support vector machine that makes fewer distinctions than a fine Gaussian SVM, using the Gaussian kernel with kernel scale set as the square root of the number of predictors. (3) Coarse KNN is a nearest-neighbor classifier that makes coarse distinctions between classes. (4) Bagged trees is a bootstrap-aggregated ensemble of fine decision trees. For each method, we used cross-validation during the training. These classifiers were implemented using the Classification Learner App [19] in MATLAB 2018a, which has the advantage of integrating multiple classifiers and eliminating the necessity to set any complex parameters.

2.5. Convolutional Neural Network. There are two ways to train a CNN: training from scratch using random numbers as starting values or using transfer learning. In many practical tasks, it is not possible to train a network from the beginning, since it takes significant amount of computing time to obtain a large number of training data and using a small amount of data to train network from the beginning will cause over-fitting of the network. This problem can be easily solved using transfer learning [20], because the network is initialized by an optimal pretraining model (trained by the ImageNet dataset [21]) and only light tuning is required during the training process.

There are two ways to use transfer learning. One is to fixate the weight parameters of certain layers and only the parameters of other layers can be changed during the training. The other is to make all the initial weight parameters

Table 2: The network structure based on VGG19, a global average pooling layer, and two dense layers were added for transfer learning. Params label means the number of parameters per layer.

No.	Layer name	Output shape	Params
1	input_1(InputLayer)	(224,224,3)	0
2	block1_conv1(Conv2D)	(224, 224, 64)	1792
3	block1_conv2 (Conv2D)	(224, 224, 64)	36928
4	block1_pool (MaxPooling2D)	(112, 112, 64)	0
5	block2_conv1 (Conv2D)	(112, 112, 128)	73856
6	block2_conv2 (Conv2D)	(112, 112, 128)	147584
7	block2_pool (MaxPooling2D)	(56, 56, 128)	0
8	block3_conv1 (Conv2D)	(56, 56, 128)	295168
9	block3_conv2(Conv2D)	(56, 56, 256)	590080
10	block3_conv3 (Conv2D)	(56, 56, 256)	590080
11	block3_conv4(Conv2D)	(56, 56, 256)	590080
12	block3_pool (MaxPooling2D)	(28, 28, 256)	0
13	block4_conv1(Conv2D)	(28, 28, 512)	1180160
14	block4_conv2 (Conv2D)	(28, 28, 512)	2359808
15	block4_conv3(Conv2D)	(28, 28, 512)	2359808
16	block4_conv4 (Conv2D)	(28, 28, 512)	2359808
17	block4_pool(MaxPooling2D)	(14, 14, 512)	0
18	block5_conv1 (Conv2D)	(14, 14, 512)	2359808
19	block5_conv2(Conv2D)	(14, 14, 512)	2359808
20	block5_conv3 (Conv2D)	(14, 14, 512)	2359808
21	block5_conv4(Conv2D)	(14, 14, 512)	2359808
22	MaxPooling2D	(7, 7, 512)	0
23	global_average_pooling2d	(512)	0
24	dense (Dense)	(1024)	525312
25	dense_1 (Dense)	(2)	2050

Table 3: The accuracy of machine learning algorithm.

Algorithm		Accuracy
Logistic regression	Logistic regression	75.7
	Quadratic SVM	77.8
	Cubic SVM	77.9
	Fine Gaussian SVM	50.1
	Medium Gaussian SVM	**79.2**
	Coarse Gaussian SVM	79.0
KNN	Fine KNN	57.3
	Medium KNN	72.4
	Coarse KNN	**78.2**
	Cosine KNN	74.8
	Cubic KNN	60.1
Ensemble	Boosted trees	78.1
	Bagged trees	**79.1**
	Subspace discriminant	78.2
	Subspace KNN	71.8
	RUSBoosted trees	75.9

alterable in the training process; this method is called fine-tuning. The first method was used in this study.

We used a shallower network VGG19 [22] as the base network. First, the weight parameters of the entire network were initialized, locking the top 22 layers, and a three-layer custom layer (a global average pooling layer and two dense layers) was added at the end of the network. The network structure is shown in Table 2. The output shape indicates width × height × dimension.

The entire network was composed of 25 layers, which included: an input layer, convolution layers, activation layers, pooling layers, and fully connected layers. The input layer accepted RGB images with a size of 224 ∗ 224. Each convolution block contained a convolution layer and an activation layer. The convolution kernels were all 3×3 in size, without padding. The ReLU [23] function was used for activation layers, and the max pooling was adopted for pooling layers. The global average pooling layer was used to prevent the entire network from overfitting. The last layer was a 2-way dense layer with softmax. The total network parameters were 20,551,746. Only the last three layers with 527,362 parameters were required to be tuned in the training. Cross entropy loss function and Adam [24] optimization algorithm were used in the model. The initial value of the learning rate was set as 0.001.

Szegedy et al. studied the effect of the depth of the network in training accuracy [25]. We also devised a study to improve the performance of the model by using a deeper network—GoogLeNet. The transfer learning was also applied to GoogLeNet in the study. The reason for choosing this network was that there were two auxiliary classifiers to compare the relationship between network depth and recognition rate. The inception structure was used in the network. The inception structure was composed of convolution kernels of 3×3, 5×5, and 1×1, respectively. The purpose was to extract richer features comparing with large convolution kernels when performing convolution operations on the same size of the receptive field [26]. The adoption of a 1×1 convolution kernel was actually a convolution operation for each pixel of the feature map. The effect was equivalent to a fully connected layer. In fact, it acted as a dimensionality reduction, and the computational complexity was significantly reduced. At the end of the structure, four convolution operations were aggregated, and different feature maps were spliced together to enrich the feature map (Hebbian principle), and another function was to decompose the sparse matrix into a dense matrix. Redundant information was not processed for the original feature map, which speeded up the calculation.

In addition to the inception structure, we also added dropout layers and randomly discarded 40% of neurons to prevent overfitting of the network. Two auxiliary classifiers were added after Inception (4a) and Inception (4d) in order to achieve better results. During the training process, the loss was added to the total loss with a certain proportion, but these were not applied to the process of inferencing. We used this structure to evaluate the effect of the depth of the network structure on the performance of the network.

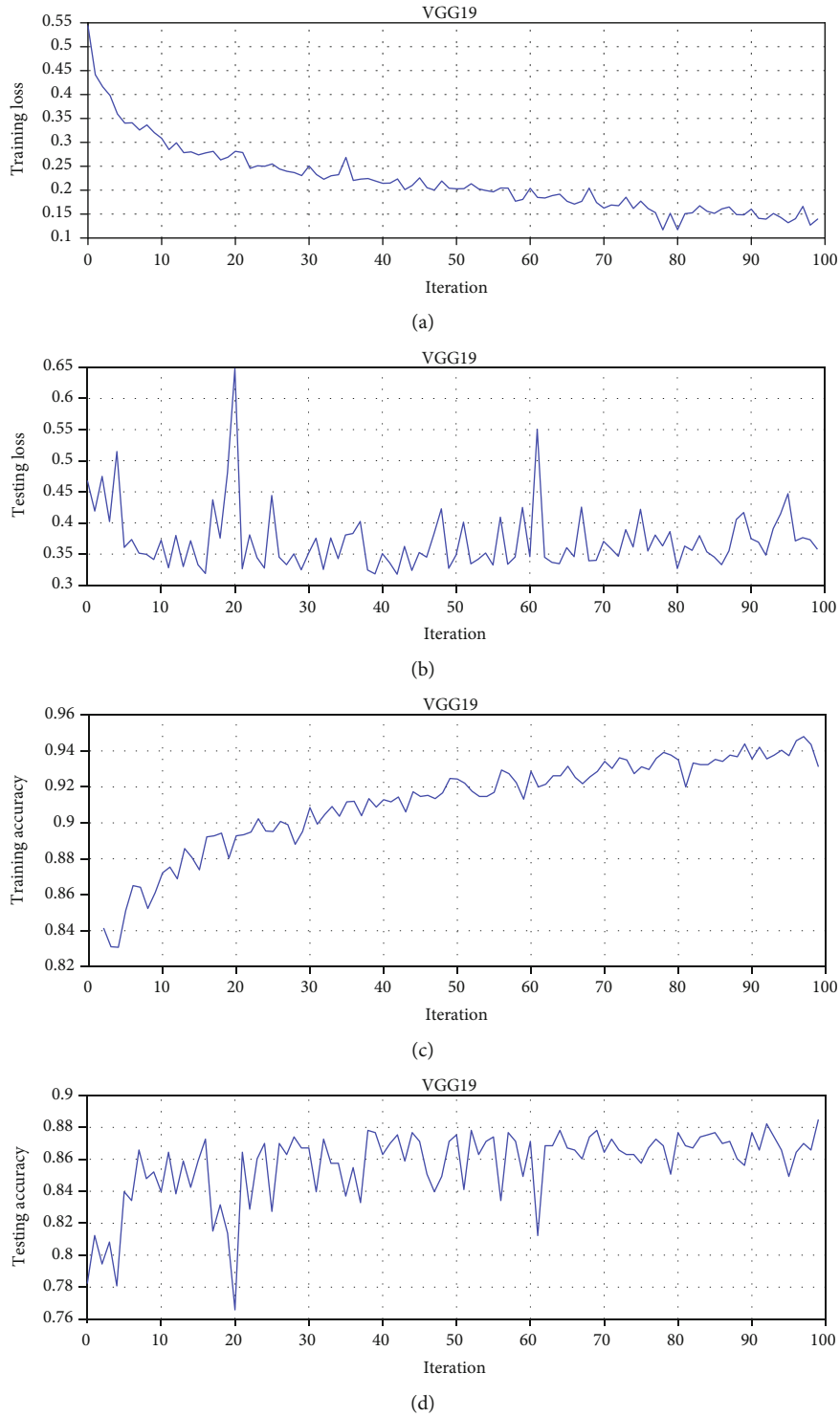


Figure 4: The curves for VGG19 during training. (a) Training loss. (b) Testing loss. (c) Training accuracy. (d) Testing accuracy.

Table 4: The confusion table of VGG19 (100 seeds were used as the validation set).

	Predict-good	Predict-bad
True-good	48	7
True-bad	8	37

3. Results

The SVM algorithm with Gaussian kernel function had the highest accuracy rate of 79.2%, which performed the best (Table 3).

We used Keras [27] with TensorFlow as the backend to get the model of VGG19. Figure 4 shows the accuracy of

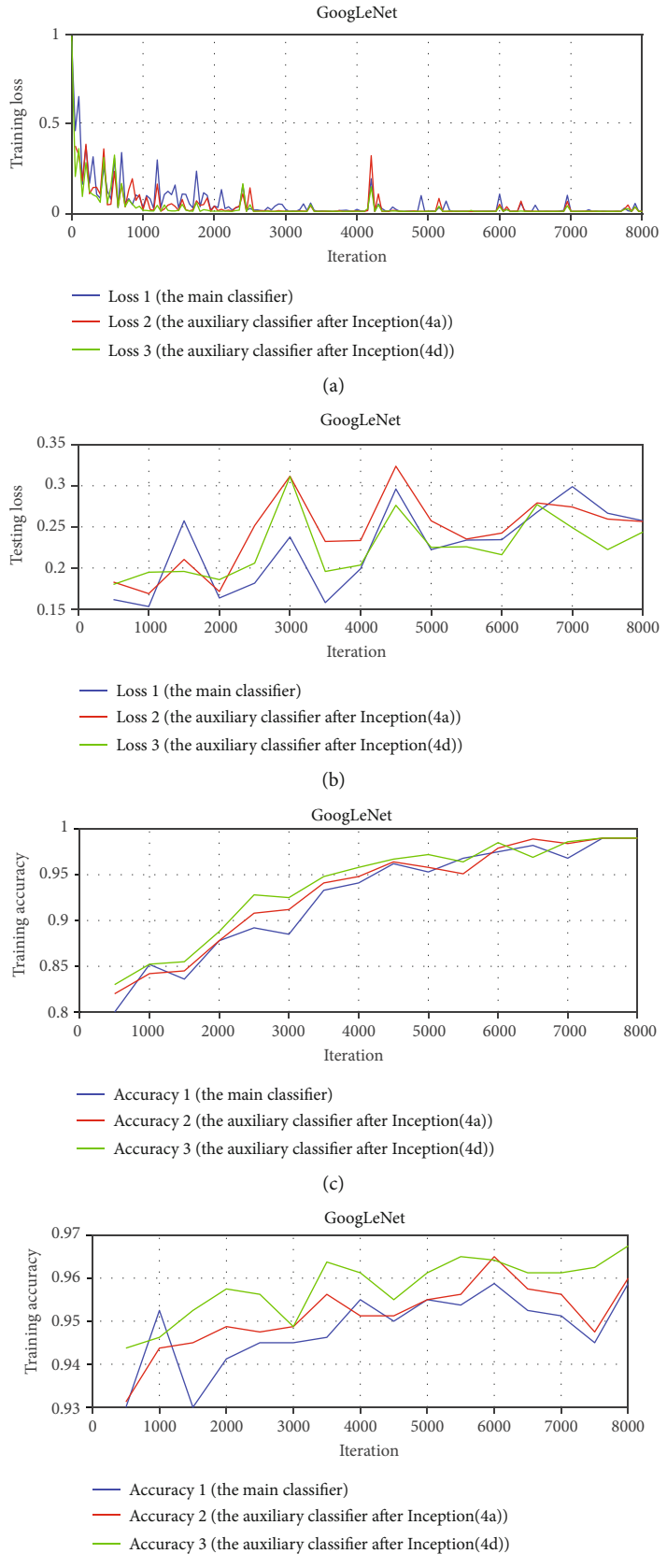


Figure 5: The curves for GoogLeNet during training. (a) Training loss. (b) Testing loss. (c) Training accuracy. (d) Testing accuracy.

Table 5: The accuracy of three classifiers (2 auxiliary classifiers) in GoogLeNet.

Layer	Test accuracy
Shallow layer	0.95754
Middle layer	0.96000
Deep layer	0.96375

Table 6: The confusion table of GoogLeNet (100 seeds were used as the validation set).

	Predict-good	Predict-bad
True-good	53	2
True-bad	4	41

training loss, testing loss, training accuracy, and testing accuracy of testing set for VGG19. The test accuracy was finally stabilized at around 88%. Comparing with machine learning algorithms, there was a significant improvement. We selected 100 maize seeds as the validation set to verify the algorithm, and the confusion table is shown in Table 4.

Since there was no integrated GoogLeNet pretraining model in Keras, we used Caffe deep learning framework. Figure 5 shows the training loss, testing loss, training accuracy, and testing accuracy of the three classifiers (the blue line and the red line represented two auxiliary classifiers, respectively, and the green line represented the main classifier). An accuracy level was reported every 500 iterations. It could be seen that the accuracy reached more than 90% in the previous iterations. Table 5 shows the accuracy of the three classifiers after 8000 iterations. The accuracy of the main classifier (deep layer) was 96.3%, and the accuracy of the auxiliary classifier (shallow layer) for the network was 95.7%. We used the same 100 seeds for algorithm verification. As shown in Table 6, the accuracy of the verification set was 94% which was slightly lower than the test set.

3.1. Visualizing Feature Maps and Heat Maps. CNNs is an end-to-end architecture. The recognition result can be automatically obtained by feeding only the pictures to be recognized to the network. The intermediate process was usually a black box and not interpretable. We used the visualization technology [28, 29] to extract the feature map of each layer in the network. In order to facilitate observation, we selected seeds with obvious damage (Figure 6(a)). Taking the first convolution layer, for example, Figure 6(b), as an output, there were 64 feature maps. It could be seen that the layer retained the original image color and texture feature information. In addition, we can observe that the features extracted by the network become more abstract as the depth of the layer increases.

We also visualized the heat map, which indicated which part of the picture was determined by the decision classification of the network. The heat map was a two-dimensional matrix that represented the CNN's score for each position of the input image. As an example, Figure 7 shows that the worm part had the greatest influence for the classification results.

4. Discussions

In the actual classification process, due to the random placement, the direction of seeds was indefinite, so the SURF (with scale and rotation invariance) algorithm was used to extract the characteristics of the seeds and there was no need to use other algorithms to correct the rotation of the seed.

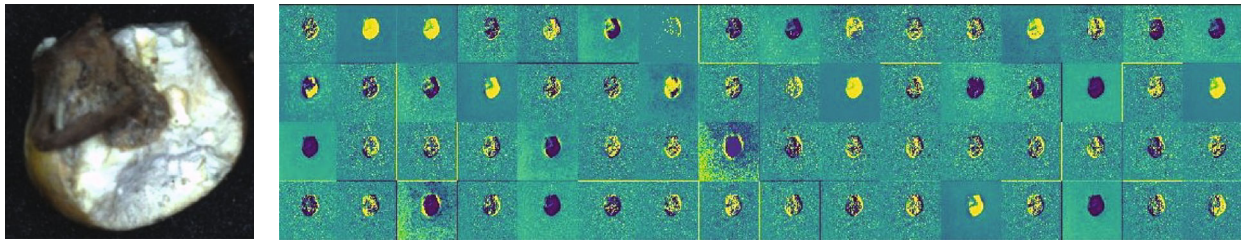
Unlike traditional machine learning algorithm, deep learning does not require complex feature extractions. We only need to consider using the appropriate network structure with the corresponding optimization algorithm. And changing the learning rate and other parameters during the training process can make the algorithm reach an optimal state. In the initial experiment, we tried to train the data from scratch using some lightweight networks (AlexNet), but these networks have experienced severe overfitting.

The reason for the excellent performance of transfer learning was that a best subset of initial values was used for model parameters. It was not necessary to start the model training with completely random numbers for all parameters. We used a shallower network (VGG19) for transfer learning: the training loss dropped significantly, but the testing loss decreased at the beginning and tended to be unchanged in the later stage. The accuracy of the training set was close to 95% and the testing accuracy of the model was stabilized at 88%, which indicated that the network had reached its optimal state. Although it had a better performance than SVM, the accuracy level was still not ideal in practice. The reason was that since VGG19 needed to identify multiple defects of seeds, the difference between some seeds with and without defects was not obvious. Therefore, the classification error was large, and it was necessary to extract deeper features. Thus, we adopted a more complex network GoogLeNet, which had an excellent ability to extract features based on the inception structure. The accuracy of the shallowest classifier was 95.7%, and the accuracy of the entire network was 96.5%. The results showed that deepness of the network played an important role. Although we achieved decent results with a deep network structure, there was overfitting during the iterative process (the training loss was decreasing and the testing loss was rising), which indicated that the amount of data available was still relatively small for complex networks like GoogLeNet.

5. Conclusions

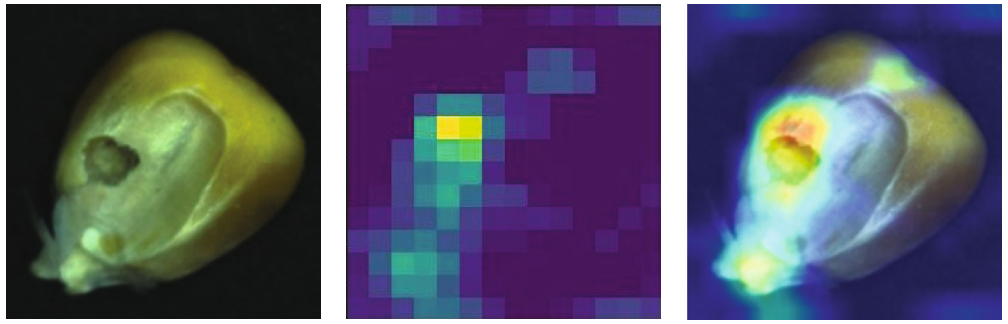
In this paper, we used CNNs and transfer learning to achieve defect classification of maize seeds. Experiments demonstrate the availability of CNN in seed defect classification tasks, the CNNs were significantly better than machine learning algorithms in maize seed defect evaluation and the accuracy of the model increased as the depth of the network increased. The appearance defect of the seed is one of the indicators for judging the quality of the seed.

In this research, we only applied the CNN to RGB images, it can also be applied to multispectral or hyperspectral images. The application of multispectral images not only has the ability to recognize the phenotypic characteristics of



(a) (b)

Figure 6: The 64 feature maps were captured from the defect seed in the first convolution layer. (a) A maize seed with defect. (b) The feature maps represent rich structure and texture feature information that expressed in a pseudo color image.



(a) (b) (c)

Figure 7: The heat map represents the contribution of each location of the input image to the network decision. (a) The original image of a worm seed. (b) The heat map of the worm seed. (c) The superposition of the original image and the heat map.

seeds but also different varieties, enhancing the generalization ability and practicability of the model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflict of interest.

Authors' Contributions

Sheng Huang and Xiaofei Fan contributed equally to this work.

Acknowledgments

This work was supported by the introduction of talent research projects in Hebei Agricultural University.

References

[1] R. Ureña, F. Rodríguez, and M. Berenguel, "A machine vision system for seeds quality evaluation using fuzzy logic," *Computers and Electronics in Agriculture*, vol. 32, no. 1, pp. 1–20, 2001.

[2] K.-l. Tu, L.-j. Li, L.-m. Yang, J.-h. Wang, and Q. Sun, "Selection for high quality pepper seeds by machine vision and classifiers," *Journal of Integrative Agriculture*, vol. 17, no. 9, pp. 1999–2006, 2018.

[3] P. M. Granitto, H. D. Navone, P. F. Verdes, and H. A. Ceccatto, "Weed seeds identification by machine vision," *Computers and Electronics in Agriculture*, vol. 33, no. 2, pp. 91–103, 2002.

[4] A. D. de Medeiros, M. D. Pereira, T. F. S. N. Soares, B. G. Noronha, and D. T. Pinheiro, "Computer vision as a complementary method to vigour analysis in maize seeds," *Journal of Experimental Agriculture International*, vol. 25, no. 5, pp. 1–8, 2018.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] Y. Lecun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[7] Y. J. Heo, S. J. Kim, D. Kim, K. Lee, and W. K. Chung, "Super-high-purity seed sorter using low-latency image-recognition based on deep learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3035–3042, 2018.

[8] B. Veeramani, J. W. Raymond, and P. Chanda, "DeepSort: deep convolutional networks for sorting haploid maize seeds," *BMC Bioinformatics*, vol. 19, no. S9, article 289, 2018.

[9] P. Ravindran, A. Costa, R. Soares, and A. C. Wiedenhoeft, "Classification of CITES-listed and other neotropical Meliaceae wood images using convolutional neural networks," *Plant Methods*, vol. 14, no. 1, article 25, 2018.

[10] L. C. Uzal, G. L. Grinblat, R. Namías et al., "Seed-per-pod estimation for plant breeding using deep learning," *Computers and Electronics in Agriculture*, vol. 14, pp. 196–204, 2018.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[12] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot multibox detector," https://arxiv.org/abs/1512.02325.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, Las Vegas, NV, USA, 2016.

[14] S. Jin, Y. Su, S. Gao et al., "Deep learning: individual maize segmentation from terrestrial lidar data using faster R-CNN and regional growth algorithms," *Frontiers in Plant Science*, vol. 9, p. 866, 2018.

[15] "Opencv-python tutorials," https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html#watershed.

[16] N. Otsu, "A threshold selection method from gray-level histogram," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[17] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," in *Proceedings of the British Machine Vision Conference 2014*, University of Nottingham, Nottingham, England, 2014.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[19] "Classification Learner APP," https://ww2.mathworks.cn/help/stats/choose-a-classifier.html.

[20] Y. Jason, C. Jeff, B. Yoshua, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems*, vol. 27, pp. 3320–3328, 2014.

[21] R. Olga, D. Jia, S. Hao et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[23] G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair," in *Proceedings of the 27th International Conference on Machine Learning*, Israel, 2010.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, San Diego, CA, USA, 2015.

[25] C. Szegedy, W. Liu, Y. Jia et al., "Going Deeper with Convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015.

[26] M. Lin, Q. Chen, and S. Yan, "Network in network," https://arxiv.org/abs/1312.4400v3.

[27] "Tensorflow-API r1.12-tf.keras," https://tensorflow.google.cn/api_docs/python/tf/keras.

[28] R. Chollet, *Deep learning with Python*, Manning Publications, 2017.

[29] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding Neural Networks through Deep Visualization," in *31st International Conference on Machine Learning*, Lille, France, 2015.