Received July 31, 2021, accepted August 12, 2021, date of publication August 16, 2021, date of current version August 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3105112

Circular Fruit and Vegetable Classification Based on Optimized GoogLeNet

FU YUESHENG[1,2], SONG JIAN[1,2], XIE FUXIANG[1,2], BAI YANG[2], ZHENG XIANG[1], GAO PENG[1], WANG ZHENGTAO[1], AND XIE SHENGQIAO[3]

[1]Shandong University of Science and Technology, Qingdao 266590, China
[2]Weifang University, Weifang 261061, China
[3]China Agricultural University, Beijing 100083, China

Corresponding author: Song Jian (sjian11@163.com)

This work was supported in part by Shandong Province Key Research and Development Program under Project 2019GNC106144.

ABSTRACT The fruit and vegetable classification problem is an inseparable branch in the field of image recognition. GoogLeNet provides a more ideal solution for the fruit and vegetable classification problem. We use the GoogLeNet network to classify apples, lemons, oranges, pomegranates, tomatoes, and colored peppers. Through experiments, we obtained the training accuracy of GoogLeNet as 96.88%, the testing accuracy as 96%, and the training speed as 11.38 sheets/second. The recognition accuracy of this model can meet the basic recognition requirements, but the training speed is low. Therefore, we decided to optimize GoogLeNet to significantly improve the training speed and further enhance the recognition accuracy of GoogLeNet. We reduced the number of convolutional kernels of GoogLeNet and adjusted the structure of Inception, which reduced the number of parameters of GoogLeNet by nearly 48% and increased the training speed of GoogLeNet from 11.38 to 33.68 sheets/second. To further improve its recognition accuracy, we tried two methods: 1) introducing a new activation function Swish; 2) between convolutional layers, we introduced DropBlock layer; these two methods improved the testing accuracy of GoogLeNet by 2%. Finally, we introduce AlexNet, VGGNet, ResNet18, DenseNet121, and Inception-ResNet to compare with our optimized GoogLeNet. By comparison, we found that our model has incredible performance in ACC, AUC, FPS, Recall, etc.

INDEX TERMS Deep learning, CNN, image classification, GoogLeNet.

I. INTRODUCTION

Iqbal *et al.* [1] proposed to achieve the classification of fruits based on color features of fruits combined with probability distribution functions; Siswantoro *et al.* [2], Rajasekar and Sharmila [3] proposed to classify fruits with the help of KNN(K-Nearest Neighbors) and color and texture features; Yang *et al.* [4], Saeed *et al.* [5] proposed to use logistic regression and inconsistent spectral wavelengths of fruits of the same species before and after ripening features to classify fruits; some other scholars use support vector machines with various features of fruits such as shape, texture, and color to classify and identify fruits, such as

Septiarini *et al.* [6], Zhang and Wu [7], Liu *et al.* [8], Lin *et al.* [9], Castro *et al.* [10], Qureshi *et al.* [11], etc.; Hasan and Monir [12], Javel *et al.* [13], Khan *et al.* [14] and others proposed fruit classification based on fuzzy logic combined with shape and color features. In addition to this, Bani and Fekri-Ershad [15] proposed to retrieve images of fruits and vegetables using color, texture features combined with spatial and frequency domains. Armi and Fekri-Ershad [16] proposed an improved texture feature extraction approach that can also be used for fruit retrieval. These traditional recognition methods not only have a low recognition accuracy of only about 90% but also these algorithms require additional features with them to be applied, so the structure is complex and difficult to implement. To meet the needs of practical production and application, new fruit recognition classification methods need to be sought.

In 2006, Hinton *et al.* proposed deep belief network (DBN), which effectively overcame the training problem of deep neural networks and greatly improved the ability of neural network feature extraction, from then on, the concept of deep learning gradually entered people's view, and the

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

ice of frozen artificial neural network gradually melted once again. In 2012, AlexNet proposed by Hinton and his student Alex completely conquered the world, from then on, deep learning (deep learning) represented by a convolutional neural network (CNN) ushered in the era of rapid development. After years of development, deep learning (DL) has derived a variety of neural networks such as convolutional neural network (CNN), recurrent neural network (RNN), generative adversarial network (GAN), and fully connected network (FCN), etc. They are widely used in image recognition, target detection, semantic image segmentation, speech recognition, natural language processing, and other fields.

With the extraordinary performance shown by CNN in many fields, many scholars began to try to apply CNN to the field of fruit recognition. CNN does not need an additional feature extraction process when performing classification recognition, and the whole feature extraction and classification process are realized in one link, so the model structure is simple and the implementation process is relatively easy, which can solve the image recognition problem well, but there are still some of the problems. Zeng [17] proposed to use VGG structure to classify 26 different vegetables and fruits with a variety of species and strong generalization with 95.6% accuracy, which meets the needs of fruit classification but is slightly inadequate in terms of training speed; Siddiqi [18] used VGG16 and InceptionV3 framework for fruit classification, respectively, and tested on the Fruit 360 dataset with high accuracy but no details on the training speed; Pande *et al.* [19] proposed to classify apples, lemons, pears, and oranges using InceptionV3 structure with accuracy close to 90% and slightly lower accuracy; Ponce *et al.* [20] used Inception-ResNetV2 model for different kinds of olive classification with the highest accuracy of 95.91%, and the model has medium accuracy; Nasiri *et al.* [21] used the VGG framework for the recognition of dates, and the model did not elaborate on the training speed although the accuracy was close to 96%. Wang *et al.* [22] proposed the internal mechanical damage detection of blueberries using ResNet and its improved version, and its detection accuracy was higher than the traditional machine learning algorithms by about 2.1%. Besides, [23]–[29] used CNN for disease detection of fruits and vegetables and achieved good results.

In summary, CNN can provide a solution to the fruit and vegetable classification problem, but the testing accuracy of current CNN models can be limited to a threshold range of 96% or the training speed is too slow. Therefore, this paper proposes to use GoogLeNet to achieve the classification of apples, lemons, oranges, pomegranates, tomatoes, and colored peppers, and optimize GoogLeNet to obtain an improved convolutional neural network for solving the problem of low training speed and improving the training accuracy and testing accuracy of the model as much as possible in the process of fruit and vegetable classification by GoogLeNet.

A. OUR CONTRIBUTIONS

Our contributions are highlighted as follows:

(1) Classification recognition of apples, lemons, oranges, pomegranates, tomatoes, and colored peppers is achieved using the GoogLeNet network.

(2) To solve the problem of the low training speed of the GoogLeNet network, we adjusted the structure of Inception and reduced the number of convolutional kernels of the model.

(3) To further improve the accuracy of GoogleNet, we tried to make two changes, and these changes improved the accuracy of the model by about 2%.

(4) Comparing our optimized GoogleNet with today's popular CNN models (ALexNet, VGG, ResNet, DenseNet, etc.) and performing model performance evaluation.

The rest of the paper is organized as follows: first, we use GoogLeNet for the classification task in Section 2 to identify the problem and propose an optimized GoogLeNet; second, we will compare the two models and give a detailed data comparison between the optimized model and other comparative models in Section 3; finally, we will give a summary and overview of the above experiments in Section 4.

II. METHODS

A. GoogLeNet

1) DETAILS OF GoogLeNet

In 2014, GoogLeNet won first place in the ImageNet competition. The structure inherits some framework structures from LeNet and AlexNet, with some changes in network depth and width. The structure has 22 network layers but only 1/36th of the parameters of VGG. It innovatively uses parallel structure, which greatly shortens the training cycle. GoogLeNet and VGGNet have pushed the research boom of deep learning to the peak.

2) EXPERIMENT OF GoogLeNet

Our experiment was carried out on a deep learning workstation. The workstation uses WIN10 Professional 64-bit operating system, the programming software is Python3.8 version, Intel(R) Xeon(R) CPU E5-2609 v4@ 1.70GHZ, RAM is 32GB, 1T SSD, and accelerated with NVIDIA GeForce GTX1080Ti. The model is implemented in the TensorFlow deep learning framework.

We use apples, colored peppers, lemons, oranges, tomatoes, and pomegranates as our experimental data. The training process of CNN is the process of repeatedly training the model to obtain the optimal weights for each class, and the whole process of CNN can be simplified into two processes: convolution and pooling. The convolution process is a process of scaling up the features, and the pooling process is a process of extracting features. These features contain color features, shape (contour) features, texture features, etc. The more the number of features differs between different kinds, the greater the difference between features of the same kind,

TABLE 1. Related parameters of the GoogLeNet model.

Type	Kernel_size /stride/ padding	Output _size	depth	1x1	3x3 (#)	3x3	5x5 (#)	5x5	Pool (#)
Convlution	7+2(s)	112x112x64	1						
MaxPool	3+2(s)	56x56x64	0						
Convlution	3+3(s)	56x56x192	2						
MaxPool	3+2(s)	28x28x192	0						
Inception （3a）		28x28x256	2	64	96	128	16	32	32
Inception (3b)		28x28x480	2	128	128	196	32	96	64
MaxPool	3+2(s)	14x14x480	0						
Inception (4a)		14x14x512	2	192	96	208	16	48	64
Inception (4b)		14x14x512	2	160	112	224	24	64	64
Inception (4c)		14x14x512	2	128	128	256	24	64	64
Inception (4d)		14x14x528	2	112	144	288	32	64	64
Inception (4e)		14x14x832	2	256	160	320	32	128	128
MaxPool	3+2(s)	7x7x832	0						
Inception (5a)		7x7x832	2	256	160	320	32	128	128
Inception (5b)		7x7x1024	2	384	192	384	48	128	128
Average Pool	7+1(s)	1x1x1024	0						
Dropout		1x1x1024	0						
linear		1x1x512	1						
softmax		1x1x6	0						

and the easier it is when using CNN for image classification. If a model can distinguish well between objects with the same shape but different colors and textures, it is perfectly capable of classifying objects with different shapes, colors, and textures. Therefore, here, we choose vegetables and fruits with the same shape as much as possible as our experimental data. The entire sample contains 6600 pictures, each fruit or vegetable contains 1100 pictures, of which 1000 are used as the training set and 100 are used as the test set. Borrowing image enhancement technology to zoom, rotate, and crop the training set, the training set is expanded to 18,000 pictures.

The model uses the Adam optimizer to update the gradient, and the learning rate is set to 0.01. When the gradient does not change after 2 epochs, the learning rate decays to the original 0.9, the epochs is set to 1000, and the batch_size is set to 64.

Through experiments, we found that the training speed of GoogLeNet is 11.38 sheets per second, and the training accuracy is 96.88%. The accuracy of the model can meet the basic recognition requirements, but the training speed is low. Therefore, we decided to optimize GoogleNet to greatly increase the training speed of GoogleNet and further enhance its accuracy.

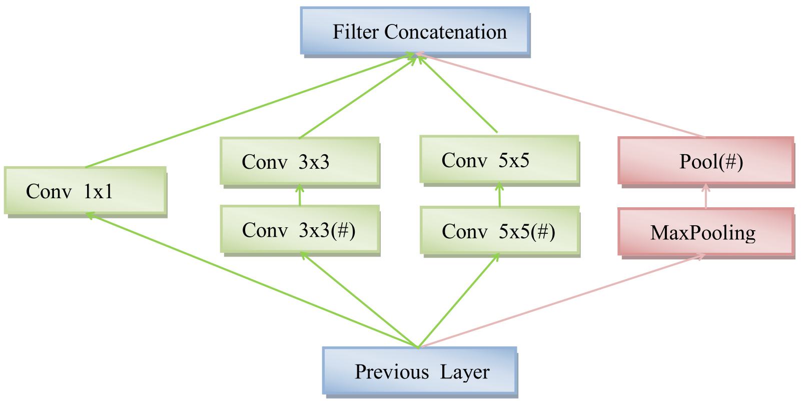


FIGURE 1. The structure of inception.

TABLE 2. Overview of the number of samples in the data.

sample	Training_ Images	Testing_ Images	Total
apple	1000*3	100	3100
color pepper	1000*3	100	3100
lemon	1000*3	100	3100
orange	1000*3	100	3100
tomato	1000*3	100	3100
pomegranate	1000*3	100	3100
Total	6000*3	600	18600

TABLE 3. Detailed data of GoogLeNet.

GoogLeNet		
ACC	LOSS	Training Speed (sheets/sec)
0.9688	0.1143	11.38

B. OPTIMIZED GoogLeNet

By understanding the existing models, we found that to improve the training speed of a model, we can achieve it by reducing the number of training parameters as much as possible while keeping the model performance. The number of training parameters is related to the size of the convolutional kernel, the number of convolutional kernels, and the depth of the network. Therefore, in order to reduce the training parameters, we fine-tune the size and number of convolutional kernels and tune the core structure of GoogLeNet, Inception. With these adjustments, we reduced the number of GoogLeNet training parameters by 48%. Besides, to further improve the training accuracy of the model, we tried to make two changes to the model: 1) Swish instead of ReLU; 2) introducing DropBlock between convolutional layers. Finally, we removed the redundant classifiers from GoogLeNet. Here is the detailed description of Optimized GoogLeNet.

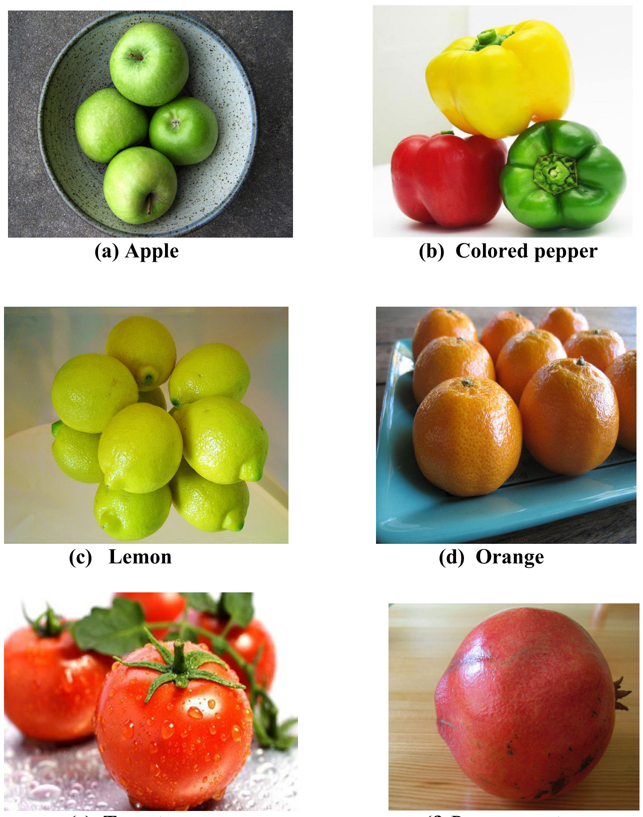


FIGURE 2. Sample example of the data.

1) DENSE-INCEPTION

Dense-Inception is an adapted Inception structure, which consists of two convolutional layers and a fully densely connected layer in parallel. In order to reduce the number of parameters, we reduce the depth of the fully dense connection, while the fully dense connection ensures the completeness of the feature information. To ensure the feature extraction capability of the network, we choose to concatenate two additional convolutional layers. The sizes of the two convolutional layers are 1×1 and 3×3. We choose different

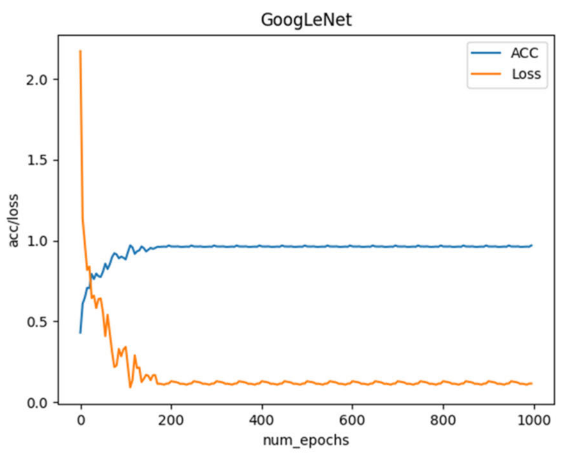


FIGURE 3. Training accuracy and loss function of GoogLeNet.

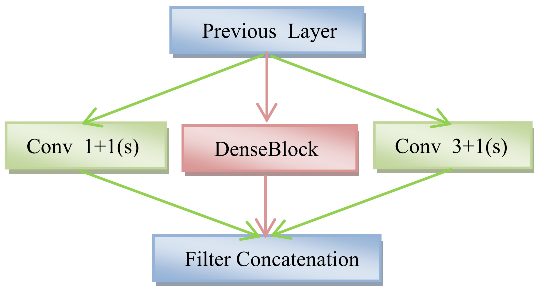


FIGURE 4. Dense-Inception structure.

sizes of convolutional kernels in order to extract features more comprehensively, and we do not choose larger kernels (e.g., 5 × 5, 7 × 7) in order to reduce the number of hidden layer neuron parameters.

We use the DenseBlock layer in DenseNet as our fully densely connected layer, but we will reduce the number of layers of the fully densely connected layer to speed up the computation and mitigate the network degradation. DenseBlock consists of multiple layers of denselayer, and the denselayer consists of 2 layers of convolution. The convolution kernels corresponding to these two layers are of 1 × 1 size, and a batch normalization layer is introduced between the convolution layer and the activation function, which has the effect of speeding up the training and solving the gradient disappearance, as well as disrupting the data set to prevent the data from being shifted. Finally, to suppress the overfitting problem of the model, a DropBlock layer needs to be introduced after each convolutional layer.

2) ACTIVATION FUNCTION

The tuning of the activation function makes one of our attempts to improve the training accuracy of the model. We need to ensure that the gradient at the infinity of the activation function is not 0 when we choose the activation function so that the activation function can converge faster and ease the gradient disappearance.

The function of ReLU is as follows:

$$f(x) = max(0, x) \tag{1}$$

Its derivative is:

$$\dot{f}(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & others \end{cases} \tag{2}$$

From the above equation, it can be seen that the derivative of the ReLU function is constantly equal to 1 in the positive half-axis of x, but constant 0 in the negative half-axis of x. Therefore, it can be seen that the ReLU function may have the problem of gradient disappearance in the negative half-axis of x.

Swish function [32] is a new activation function proposed by GoogLe in 2017. Several experiments in [32] point out that the gradient disappearance of the network model using the ReLU function becomes more and more obvious as the

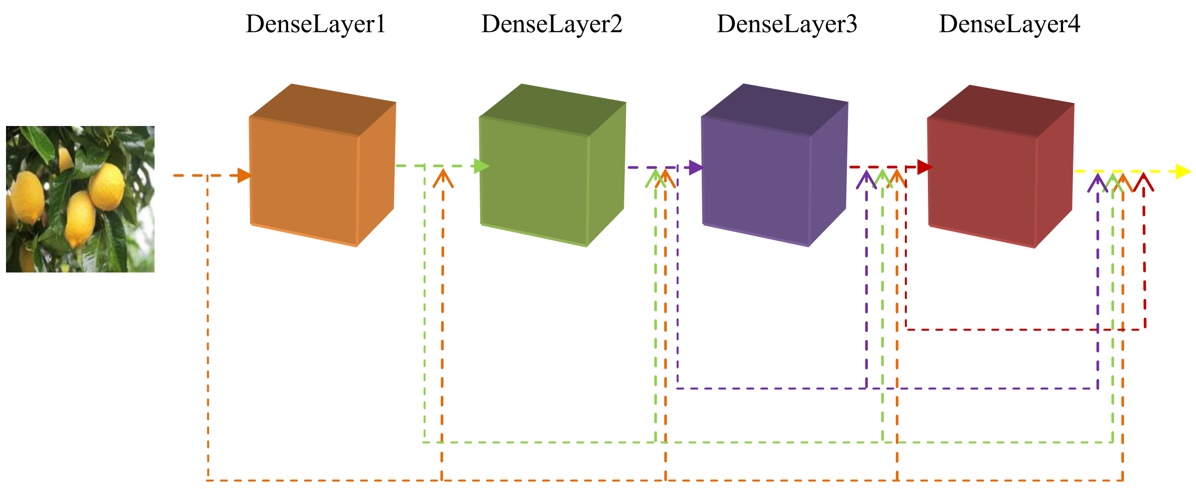


FIGURE 5. DenseBlock structure.

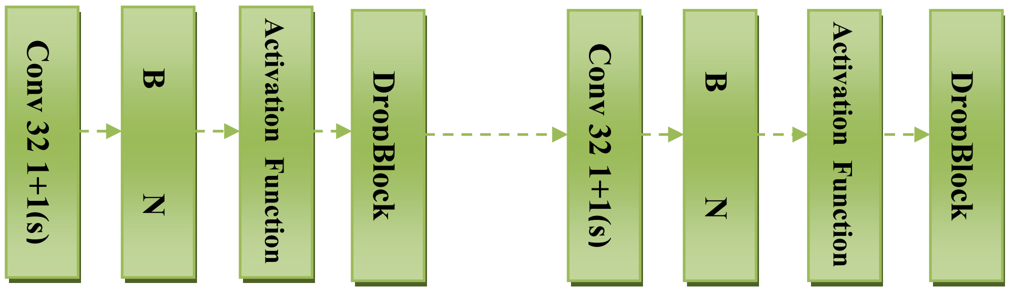


FIGURE 6. DenseLayer structure.

depth of the network increases, but the Swish function still has very excellent convergence performance.

The function of Swish is as follows:

$$g(x) = x^* sigmoid(\beta x), x \in (-\infty, +\infty) \tag{3}$$

Swish is a kind of lower bound without upper bound, smooth non-monotonic function, according to the different values of β, Swish function exhibits different properties.

When $\beta = 0$, $g(x) = \frac{x}{2}$. its output range is $(-\infty, +\infty)$. When $\beta \rightarrow \infty$, $g(x)$ becomes a function similar to ReLU, and its output is max(0, x). When $\beta = 1$, $g(x) = x^* sigmoid(x)$, which outputs in the range $(-0.5, +\infty)$. The default here is =1. If $u(x) = sigmoid(x)$, then $g(x) = x^* u(x)$. Its derivative is:

$$\dot{g}(x) = x^* \dot{u}(x) + u(x) \tag{4}$$

$$\dot{g}(x) = g(x) + u(x)(1 - g(x)) \tag{5}$$

From Eq. (5), we can find that the derivative of the Swish function at infinity is not zero whether x takes positive or negative values. therefore, in a sense, the convergence performance of the Swish function is stronger than that of ReLU.

3) REGULARIZATION – DROPBLOCK

Introducing DropBlock between the convolutional layers is another attempt we made to improve the training accuracy of the model. To suppress the overfitting ability of the model, speed up the training speed and reduce the number of neurons, our common method is to introduce the dropout layer: when training the model, some neurons in the hidden layer are randomly removed, and the output is multiplied by the same proportion. dropout layer has a very obvious effect in the fully connected layer, but its powerful ability is not reflected in the convolutional layer, because the features on the convolutional layer are correlated, and even if the dropout layer is applied, the relevant information will still be passed to the next layer, as shown in Figure 7(a). Therefore, we need a spatial dropout layer. the emergence of DropBlock [33] solves the problem of spatial deletion on convolutional layers, as shown in Figure 7(b). It can preserve the features of the image more comprehensively when performing large deletion of neurons because it is overall block deletion rather than random deletion.

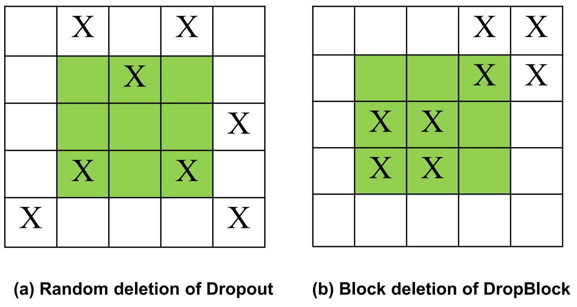

X
(a) Random deletion of Dropout
(b) Block deletion of DropBlock

FIGURE 7. The work way of dropout and DropBlock.

The DropBlock takes two main arguments, *block_size* and γ. The *block_size* indicates the size of the dropout block ("X" in Figure 7(b)), which is usually set to a fixed value. When *block_size* = 1, DropBlock degenerates to the traditional dropout, which can normally be taken as 3, 5, 7. According to the experimental data mentioned in [33], when *block_size*=7, the experimental data is more ideal, so we take 7 for block_size here. γ represents the number of hidden layer deletion activation units.

$$\gamma = \frac{1 - k\_prob}{block\_size^2} * \frac{f\_size^2}{(f\_size - block\_size + 1)^2} \tag{6}$$

where, *f_size* represents the size of the feature graph. *k_prob* represents the probability that we keep each activation unit; According to the experimental data mentioned in [33], a linearly decreasing value of *k_prob* will result in better test accuracy, so here we take 0.95 - 0.8 for *k_prob* in a linearly decreasing manner. starting from the second downsampling pooling layer, *k_prob* decreases by 0.05 for each downsampling pooling layer passed. Refer to Table 4 for specific values.

4) THE OVERALL STRUCTURE OF THE OPTIMIZED GoogLeNet

In summary, the structure of the optimized GoogLeNet contains 15 layers, including 2 convolutional layers, 5 pooling layers, 6 Dense-Inception layers, and 2 fully connected layers. Figure 8 shows the general structure of the Optimized GoogLeNet. Table 4 shows the specific parameters related to Optimized GoogLeNet. By reducing the number of

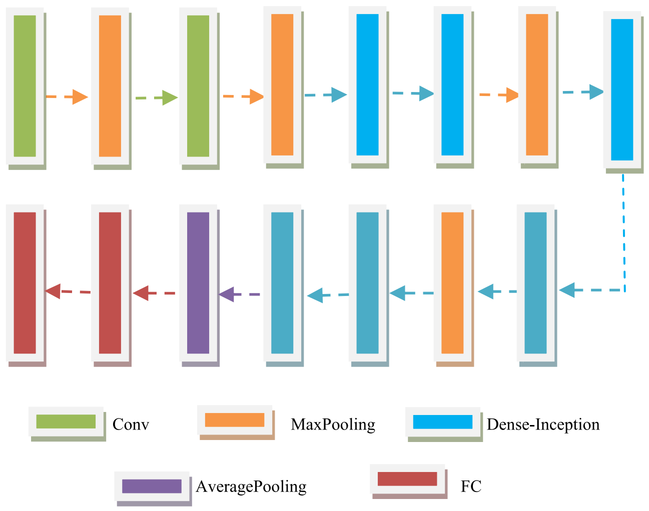


FIGURE 8. Schematic diagram of Optimized GoogLeNet structure.

convolutional kernels and changing the Inception structure, we reduced the number of training parameters of GoogLeNet by 3 million.

III. EXPERIMENT

In this section, we perform the model evaluation of our model and compare the relevant parameters with the mainstream models nowadays. This time, we used five quantitative methods to evaluate the model, namely Accuracy, Recall, Precision, AUC, and F1_score. They are calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$F1\_score = 2^{*} \frac{Precision^{*} Recall}{Precision + Recall} \tag{10}$$

where *TP, TN, FP* and *FN* are True Positive, True Negative, False Positive and False Negative. The value of AUC is the area under the ROC curve.

A. GoogLeNet VERSUS OPTIMIZED GoogLeNet

In this section, we will use GoogLeNet to compare the performance with our optimized GoogLeNet. All the above models

TABLE 4. Related parameters of the optimized GoogLeNet model.

Name	Type	filters	Kernel _size/ stride/ padding	Output _size	depth	1x1	3x3	(Dense -layer) 1x1	params	k_prob
Conv1	Convlution /BN/DB	32	7+2(s)	112x112x32	1				4K	0.95
maxpool1	MaxPool	32	3+2(s)	56x56x32	0					
Conv2	Convlution /BN/DB	32	1+1(s)	56x56x32	1				1K	0.95
Maxpool2	MaxPool	32	3+2(s)	28x28x32	0					
D_I(32,2,32)	Dense-Inception			28x28x128	2	32	32	64	16K	0.9
D_I(32,2,32)	Dense-Inception			28x28x128	2	32	32	64	49K	0.9
Maxpool3	MaxPool	128	3+2(s)	14x14x128	0					
D_I(64,4,64)	Dense-Inception			14x14x256	4	64	64	128	245K	0.85
D_I(64,4,64)	Dense-Inception			14x14x256	4	64	64	128	491K	0.85
Maxpool4	MaxPool	256	3+2(s)	7x7x256	0					
D_I(128,8,128)	Dense-Inception			7x7x512	4	128	128	256	980K	0.8
D_I(128,8,128)	Dense-Inception			7x7x512	4	128	128	256	1900K	0.8
averagepool	AveragePool	512	7+1(s)	1x1x512	0					
FC1(linear)	Dense/dp(0.6)	512		1x1x512	1				260K	
FC2(softmax)	Dense	6		1x1x6	0					

TABLE 5. Schematic diagram of confusion matrix.

Predict \ Actual	1	0
1	True Positive	False Positive
0	False Negative	True Negative

use Adam optimizer for gradient update, the learning rate is set to 0.01, and when the gradient does not change after every 2 epochs, the learning rate decays to the original 0.9, the epochs are set to 1000, and the batch_size is set to 64. Due to our limited experimental data, in order to improve the performance of the optimized model, we first pre-trained our model with the ImageNet2012 dataset to obtain the initialized weight parameters and then retrained the model with our experimental data to obtain the optimal weights.

1) SWISH AND DROPBLOCK ON GoogLeNet

In this experiment, we monitor the effect of Swish and DropBlock on GoogLeNet by the control variables method. To compare the performance of Swish and ReLU functions, we introduced the two functions into GoogLeNet separately with dropout between convolutional layers, and the comparison results are shown in Table 6. The experimental table Swish function improves the accuracy by 0.14% over the ReLU function.

TABLE 6. Comparison of different activation functions.

Function	ACC	Error Rate
ReLU	96.88%	3.12%
Swish	97.02%	2.98%

To compare the performance of DropBlock and dropout, we introduce these two layers between the convolutional layers respectively and choose ReLu for the activation function. Through Table 7 we see that the effect of adding DropBlock layer between the convolutional layers is better than that of introducing dropout, and the accuracy is improved by 0.33% after introducing DropBlock.

TABLE 7. Comparison of DP and DB.

Function	ACC	Error Rate
DropBlock	97.21%	2.97%
dropout	96.88%	3.12%

Finally, we introduce Swish and DropBlock into GoogLeNet at the same time and get the following data. We find that the training accuracy of GoogLeNet is improved by 0.71% after introducing DB and Swish. This shows that our two attempts can improve the training accuracy of GoogLeNet.

TABLE 8. Comparison of parameters after precision optimization.

Function	ACC	Error Rate
DB+Swish	97.59%	2.41%
dp+ReLU	96.88%	3.12%

2) COMPARISON OF TRAINING SPEED

In this section we compare the structure of GoogLeNet with the optimized speed, we use the control variables method, we choose ReLU for the activation function, and we choose dropout between the convolutional layers. through the experiments, we find that the training speed of GoogLeNet is significantly improved after the speed optimization, almost three times as fast as the unoptimized one. Therefore, our speed optimization method does have the effect of improving the training speed.

TABLE 9. Parameter comparison after speed optimization.

Function	FPS
Before	11.38
After	33.68

3) GoogLeNet VERSUS OPTIMIZED GoogLeNet

In this section, we compare GoogLeNet with our optimized GoogLeNet, and we find that the training accuracy of the optimized model is 98.82%, which is a 1.94% improvement over GoogLeNet. This training accuracy is much higher than the two attempts we made to improve the training accuracy. the Dense-Inception structure uses a shallow depth in order to reduce the number of neurons, and to ensure the comprehensiveness of feature extraction, we use the aid of fully dense connections and two convolutional kernels of different sizes. This design makes our model feature extraction more powerful and, perhaps, is the reason for the greater improvement in the training accuracy of our model.

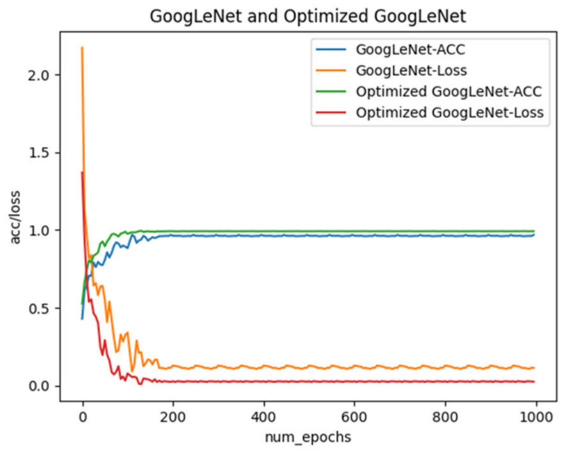


FIGURE 9. Comparison of ACC and Loss about GoogLeNet and Optimized GoogLeNet.

GoogLeNet and Optimized GoogLeNet were tested separately using the test set. There are 600 images in the test set,

TABLE 10. GoogLeNet versus Optimized GoogLeNet regarding training parameters.

Model	ACC	Loss	Error Rate	Fps
GoogleNet	96.88%	0.1143	3.12%	11.38
Optimized GoogLeNet	98.82%	0.0254	1.18%	33.68

TABLE 11. Test accuracy comparison between GoogLeNet and optimized GoogLeNet.

Model	ACC	Error Rate
GoogleNet	96%	4%
Optimized GoogLeNet	98%	2%

and 100 images in each class. Through the tests, we found that the testing accuracy of Optimized GoogLeNet improved by 2% compared with GoogLeNet.

B. COMPARISION OF OPTIMIZED GoogLeNet WITH OTHER MODELS

In this section, we will choose AlexNet, VGGNet, ResNet18, DenseNet121, and Inception-ResNet to compare with our optimized GoogLeNet.

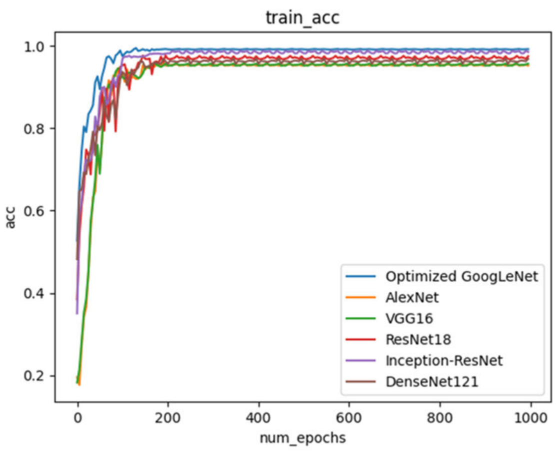


FIGURE 10. Comparison of training accuracy of each model.

Figure 10, Figure 11, and Table 12 show the detailed comparison of the performance parameters of each model. Figure 10 shows the comparison of the training loss function of each model, Figure 11 shows the comparison of the training accuracy of each model, and Table 12 shows the detailed data of the accuracy, Error Rate, and Fps of each model. Figure 11 shows that Optimized GoogLeNet can stabilize faster than other models. In terms of training accuracy Optimized GoogLeNet has almost the highest accuracy, 98.82%, which is 0.28% higher than that of Inception-ResNet, followed by ResNet18, DenseNet121, GoogleNet, VGG16, and AlexNet. in terms of training speed ResNet18 has an fps of

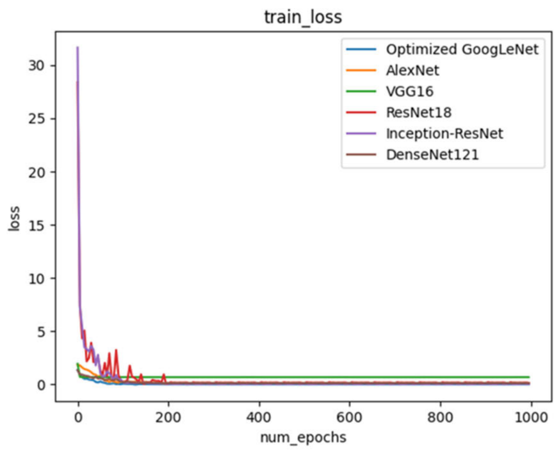


FIGURE 11. Comparison of loss of each model.

TABLE 12. Comparison of different models on ACC, FPS.

Model	ACC	Loss	Error Rate	Fps
AlexNet	95.25%	0.1421	4.75%	9.0
VGGNet16	95.69%	0.6933	4.43%	6.09
ResNet18	97.51%	0.1522	2.49%	39.78
DenseNet121	97.08%	0.1276	2.92%	15.61
InceptionResNet	98.54%	0.0509	1.46%	29.77
Optimized GoogLeNet	98.82%	0.0254	1.18%	33.68

39.78, which is the fastest running speed among the tested models, and our model has an fps of 33.68, which ranks the second among the tested models. In summary, our optimized model performs better among many mainstream models, is comparable to Inception-ResNet in terms of training accuracy, and is only slightly lower than ResNet18 in terms of training speed.

C. EVALUATION OF THE OPTIMIZED GoogLeNet

TABLE 13. Accuracy evaluation of different samples.

Sample	Recall	Precision	Specificity	AUC	F1_score
apple	0.99	0.9802	0.996	0.98	0.9851
color pepper	0.98	0.98	0.996	0.96	0.98
lemon	0.96	0.9796	0.996	0.95	0.9697
orange	0.96	0.96	0.992	0.95	0.96
pomegranate	0.97	0.97	0.994	0.96	0.97
tomato	0.99	0.9802	0.996	0.97	0.9851

For this experiment, we evaluated Optimized GoogLeNet in five aspects, including ACC, Recall, Precision, AUC, and F1-score. We evaluated the test set using the weight parameters obtained from the training set to obtain the confusion matrix [Figure 12] as well as the ROC graph [Figure 13]. Through the above experimental data, we see that

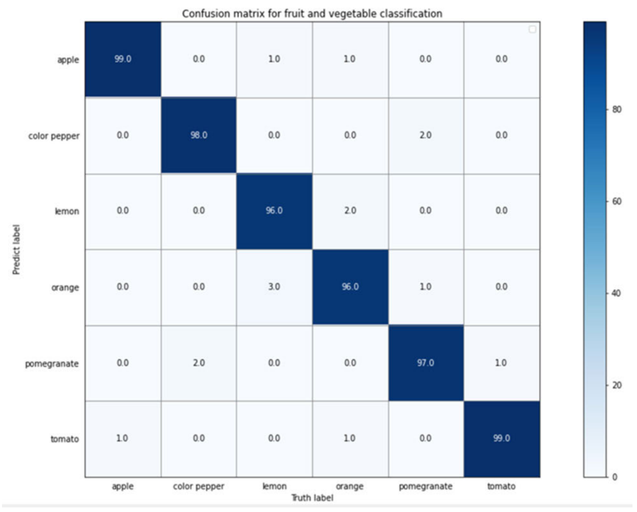


FIGURE 12. **Confusion matrix for fruit and vegetable classification.**

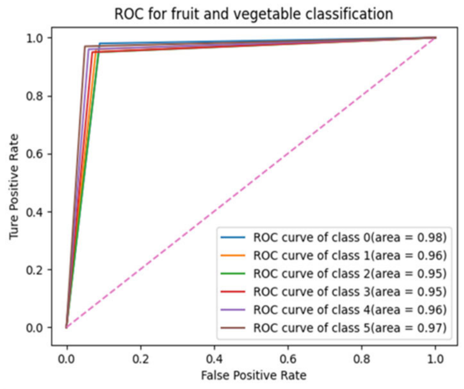


FIGURE 13. **ROC for fruits and vegetables classification.**

the F1_score of apples and tomatoes is 0.9851 and the AUC is 0.98 and 0.97, while the F1_score and AUC of lemons and oranges are above 0.95 although they are the lowest. Therefore, our optimized GoogLeNet can solve the vegetable and fruit classification problem well.

D. SINGLE IMAGE TEST OF OPTIMIZED GoogLeNet

In this section, we divide the single image test into three parts, random image test, real-time detection of images, and limitation analysis of the model.

1) RANDOM IMAGE TEST

This section tests the classification ability of the model for each category in different contexts. The images we used are mainly pictures of ripe fruits placed in different backgrounds (baskets or plates), which are randomly selected from the test set. Through the test, we found that the accuracy of all kinds of samples is very high, and the accuracy of all kinds of images reached 0.99, especially for lemons and oranges, which are very similar in shape, color, and volume, but still have very high recognition accuracy.

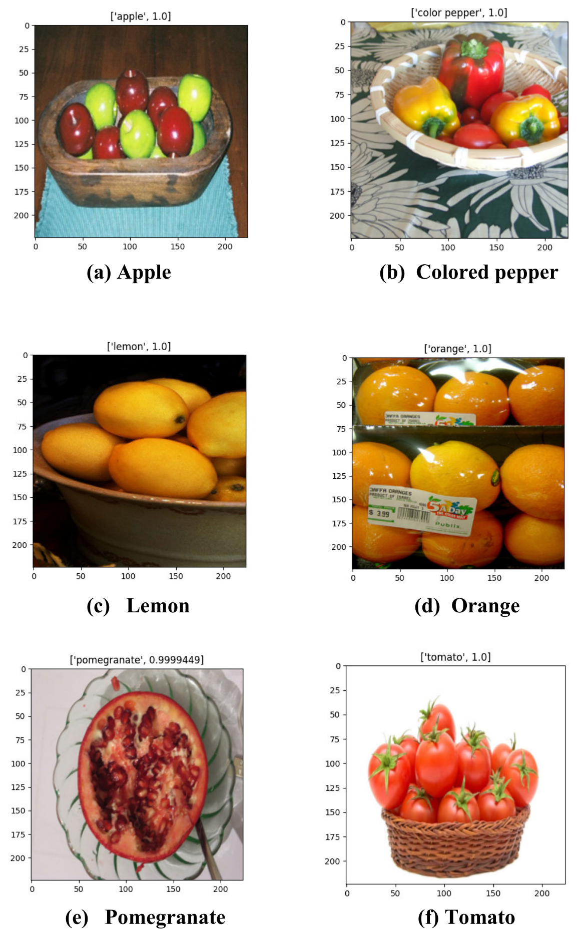


FIGURE 14. **Prediction of a random image.**

2) REAL-TIME DETECTION OF IMAGES

This section tests the classification ability of the model in each classification real-time environment. The images we chose are mainly ripe but unpicked fruits, which are selected from the test set. Through the test, we found that our optimized model still performs well in the real-time environment, and the test accuracy for each classification is above 0.97. This indicates that our model can meet the demand of real-time classification, which is helpful for us to extend the model to the field of target detection later.

3) ANALYSIS OF THE LIMITATIONS OF THE MODEL

We added this section to better understand the priority of the optimized model for different feature responses and to facilitate our understanding of how the model works. We collected or synthesized some single images to contain multiple target categories. By testing these images we found that for different objects with similar shapes, the CNN model classifies the objects using the color values of the color features [Table 14], i.e., the convolution and pooling process is the process of cumulative expansion and extraction of the color values. If two objects in the same frame have the same shape but different colors, the model will preferentially detect

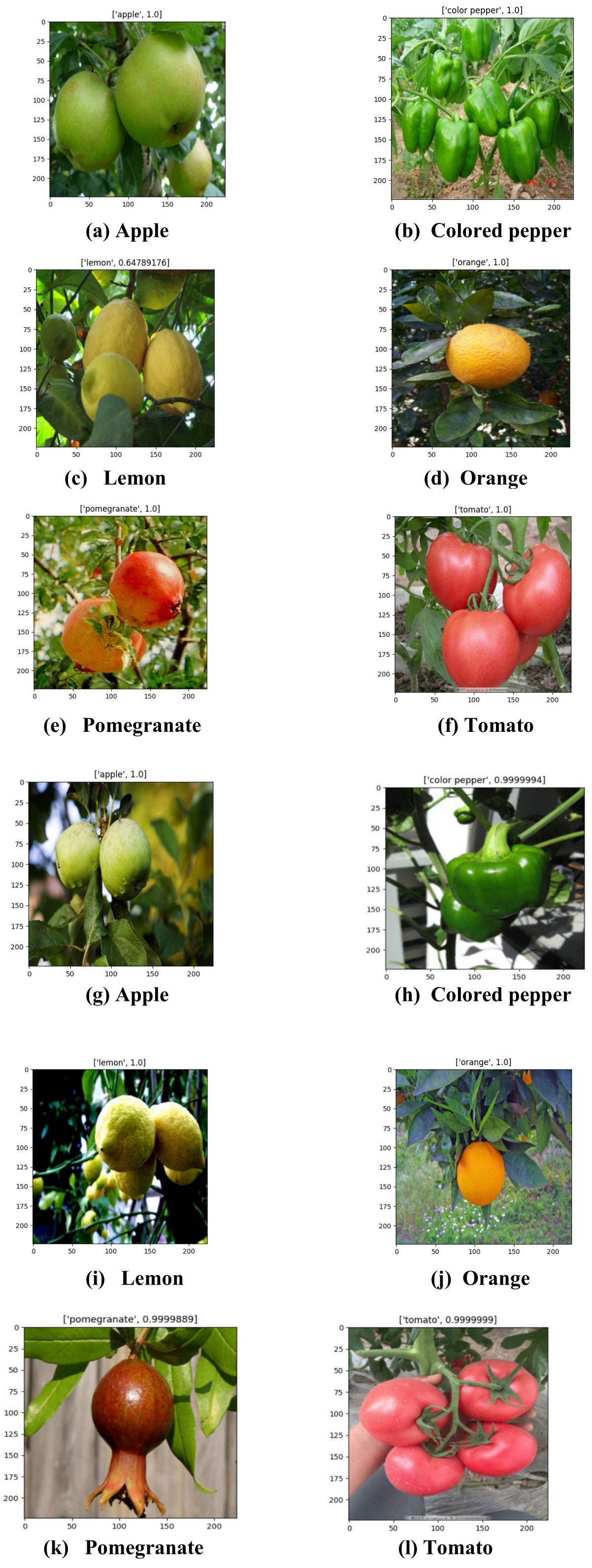


FIGURE 15. The effect of real-time detection of images.

the object with the larger color value, as in Figure 16(a-c). If the shapes are similar between different objects and the overall difference in color values is not obvious, what plays a decisive role is the texture features on the surface of the target object constructed from the color values, and the model will preferentially predict the object with more obvious texture features as in Figure 16(d-f).

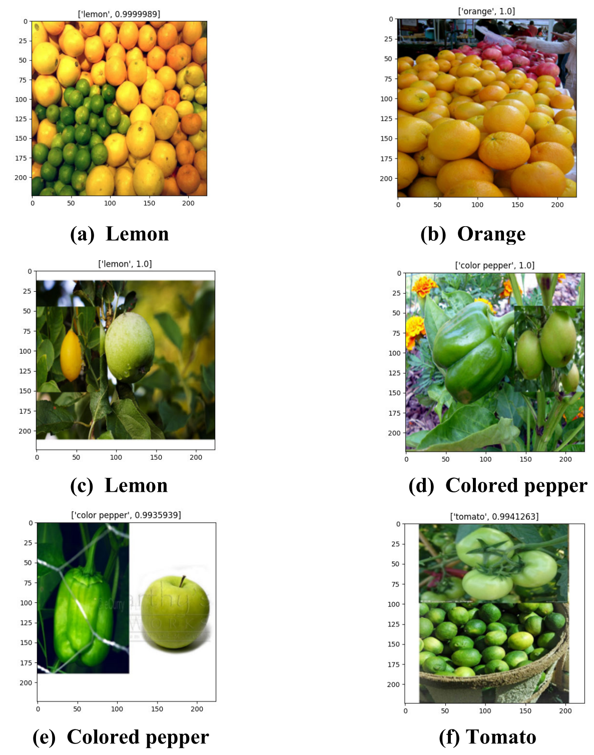


FIGURE 16. Examples of model limitations analysis.

TABLE 14. Different colors of the color value comparison table.

Color	Color Value(16)	Color Value(rgb)
Yellow	#FFFF00	(255,255,0)
Orange	#FFA500	(255,165,0)
Red	#FF0000	(255,0,0)
Green	#00FF00	(0,255,0)

In short, including shape features that are not relevant to this paper, whatever feature its decisive role is ultimately determined by the color's color value.

IV. DISCUSSION

Fruit and vegetable classification is an important application branch of deep learning in the field of image recognition. We implemented GoogLeNet to classify apples, colored peppers, lemons, oranges, pomegranates, and tomatoes. Through experiments, we found that GoogLeNet can solve this problem well, and its training accuracy reaches 96.88% and testing accuracy reaches 96.00%, which can basically meet the practical needs. However, we found that the training period of

GoogLeNet is very long during the experiment, so we decided to optimize the model and reduce the training time of the model as much as possible under the premise of guaranteeing the testing accuracy of the model. Besides, we made two attempts to improve the accuracy of the model.

(1) The training time of the model is related to the model training parameters, while the number of training parameters is related to the size of the convolutional kernel, the number of convolutional kernels, and the depth of the network. Therefore, to reduce the training parameters, we fine-tune the size and number of convolutional kernels and adjust the core structure of GoogLeNet, Inception. With these adjustments, we reduce the number of GoogLeNet training parameters by 48%. Through experiments, we find that the training speed is significantly improved after speed optimization of GoogLeNet, almost three times as fast as before optimization, with a training speed of 33.68 sheets/sec. This shows that the training speed of the model can be significantly improved by reducing the number of convolutional kernels and adjusting the Inception structure.

(2) To further improve the training accuracy of the model, we tried to make two changes to the model: 1) Swish instead of ReLU; 2) DropBlock was introduced between the convolutional layers. Finally, we removed the redundant classifiers in GoogLeNet. By controlling the variable method, we found that the training accuracy of the model improved by 0.71% after introducing Swish and DropBlock.

(3) By comparing GoogLeNet with the optimized GoogLeNet, we find that the optimized model has superior performance than GoogLeNet with 98.82% training accuracy, 1.94% improvement in training accuracy, 2% improvement in testing accuracy, and 33.68 sheets/sec training speed.

(4) We evaluate Optimized GoogLeNet in five aspects, including ACC, Recall, Precision, AUC, and F1-score. Through the experiment, we see that the F1_score of apple and tomato is 0.9851, and the AUC is 0.98 and 0.97, while the F1_score and AUC of lemon and orange are above 0.95 although they are the lowest. Therefore, our optimized GoogLeNet can solve the vegetable and fruit classification problem well.

(5) Finally, we tested the optimized GoogLeNet on a single image and found that the results were satisfactory.

REFERENCES

[1] S. M. Iqbal, A. Gopal, P. E. Sankaranarayanan, and A. B. Nair, "Classification of selected citrus fruits based on color using machine vision system," *Int. J. Food Properties*, vol. 19, no. 2, pp. 272–288, Feb. 2016.

[2] J. Siswantoro, H. Arwoko, and M. Widiasri, "Indonesian fruits classification from image using MPEG-7 descriptors and ensemble of simple classifiers," *J. Food Process Eng.*, vol. 43, no. 7, Mar. 2020, Art. no. e13414.

[3] L. Rajasekar and D. Sharmila, "Performance analysis of soft computing techniques for the automatic classification of fruits dataset," *Soft Comput.*, vol. 23, no. 8, pp. 2773–2788, Apr. 2019.

[4] C. Yang, W. S. Lee, and J. G. Williamson, "Classification of blueberry fruit and leaves based on spectral signatures," *Biosyst. Eng.*, vol. 113, no. 4, pp. 262–351, Dec. 2012.

[5] O. M. B. Saeed, S. Sankaran, A. R. M. Shariff, H. Z. M. Shafri, R. Ehsani, M. S. Alfatni, and M. H. M. Hazir, "Classification of oil palm fresh fruit bunches based on their maturity using portable four-band sensor system," *Comput. Electron. Agricult.*, vol. 82, pp. 55–60, Mar. 2012.

[6] A. Septiarini, H. Hamdani, H. R. Hatta, and A. A. Kasim, "Image-based processing for ripeness classification of oil palm fruit," in *Proc. 5th Int. Conf. Sci. Inf. Technol. (ICSITech)*, Oct. 2019, pp. 23–26.

[7] Y. Zhang and L. Wu, "Classification of fruits using computer vision and a multiclass support vector machine," *Sensors*, vol. 12, no. 9, pp. 12489–12505, Sep. 2012.

[8] X. Liu, W. Jia, C. Ruan, D. Zhao, Y. Gu, and W. Chen, "The recognition of apple fruits in plastic bags based on block classification," *Precis. Agricult.*, vol. 19, no. 4, pp. 735–749, Aug. 2018.

[9] G. Lin, Y. Tang, X. Zou, J. Xiong, and Y. Fang, "Color-, depth-, and shape-based 3D fruit detection," *Precis. Agricult.*, vol. 21, no. 1, pp. 1–17, Feb. 2020.

[10] W. Castro, J. Oblitas, M. De-la-Torre, C. Cotrina, K. Bazán, and H. Avila-George, "Classification of cape gooseberry fruit according to its level of ripeness using machine learning techniques and different color spaces," *IEEE Access*, vol. 7, pp. 27389–27400, 2019.

[11] W. S. Qureshi, A. Payne, K. B. Walsh, R. Linker, O. Cohen, and M. N. Dailey, "Machine vision for counting fruit on mango tree canopies," *Precis. Agricult.*, vol. 18, no. 2, pp. 224–244, Apr. 2017.

[12] R. Hasan and S. M. G. Monir, "Fruit maturity estimation based on fuzzy classification," in *Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA)*, Kuching, Malaysia, Sep. 2017, pp. 27–32.

[13] I. M. Javel, A. A. Bandala, R. C. Salvador, R. A. R. Bedruz, E. P. Dadios, and R. R. P. Vicerra, "Coconut fruit maturity classification using fuzzy logic," in *Proc. IEEE 10th Int. Conf. Humanoid, Nanotechnol., Inf. Technol., Commun. Control, Environ. Manage. (HNICEM)*, Nov. 2018, pp. 1–6.

[14] A. Khan, J. P. Li, R. A. Shaikh, and I. Khan, "Vision based classification of fresh fruits using fuzzy logic," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop.*, Mar. 2016, pp. 3926–3932.

[15] N. T. Bani and S. Fekri-Ershad, "Content-based image retrieval based on combination of texture and colour information extracted in spatial and frequency domains," *Electron. Library*, vol. 37, no. 4, pp. 650–666, Aug. 2019.

[16] L. Armi and S. Fekri-Ershad, "Texture image Classification based on improved local Quinary patterns," *Multimedia Tools Appl.*, vol. 78, no. 14, pp. 18995–19018, 2019, doi: 10.1007/s11042-019-7207-2.

[17] Z. Guoxiang, "Fruit and vegetables classification system using image saliency and convolutional neural network," in *Proc. IEEE 3rd Inf. Technol. Mechatronics Eng. Conf.*, Oct. 2017, pp. 613–617.

[18] R. Siddiqi, "Effectiveness of transfer learning and fine tuning in automated fruit image classification," in *Proc. 3rd Int. Conf. Deep Learn. Technol. (ICDLT)*, Xiamen, China, 2019, pp. 91–100.

[19] A. Pande, M. Munot, R. Sreeemathy, and R. V. Bakare, "An efficient approach to fruit classification and grading using deep convolutional neural network," in *Proc. IEEE 5th Int. Conf. for Converg. Technol. (I2CT)*, Mar. 2019, pp. 1–7.

[20] J. M. Ponce, A. Aquino, and J. M. Andujar, "Olive-fruit variety classification by means of image processing and convolutional neural networks," *IEEE Access*, vol. 7, pp. 147629–147641, 2019.

[21] A. Nasiri, A. Taheri-Garavand, and Y.-D. Zhang, "Image-based deep learning automated sorting of date fruit," *Postharvest Biol. Technol.*, vol. 153, pp. 133–141, Jul. 2019.

[22] Z. Wang, M. Hu, and G. Zhai, "Application of deep learning architectures for accurate and rapid detection of internal mechanical damage of blueberry using hyperspectral transmittance data," *Sensors*, vol. 18, no. 4, p. 1126, Apr. 2018.

[23] S. Xing, M. Lee, and K.-K. Lee, "Citrus pests and diseases recognition model using weakly dense connected convolution network," *Sensors*, vol. 19, no. 14, p. 3195, Jul. 2019.

[24] J.-R. Xiao, P.-C. Chung, H.-Y. Wu, Q.-H. Phan, J.-L.-A. Yeh, and M. T.-K. Hou, "Detection of strawberry diseases using a convolutional neural network," *Plants*, vol. 10, no. 1, p. 31, Dec. 2020.

[25] J. G. A. Barbedo, "Plant disease identification from individual lesions and spots using deep learning," *Biosystems Eng.*, vol. 180, pp. 96–107, Apr. 2019.

[26] J. Shin, Y. K. Chang, B. Heung, T. Nguyen-Quang, G. W. Price, and A. Al-Mallahi, "A deep learning approach for RGB image-based powdery mildew disease detection on strawberry leaves," *Comput. Electron. Agricult.*, vol. 183, Apr. 2021, Art. no. 106042.

[27] H. Cecotti, A. Rivera, M. Farhadloo, and M. A. Pedroza, "Grape detection with convolutional neural networks," *Expert Syst. Appl.*, vol. 159, Nov. 2020, Art. no. 113588.

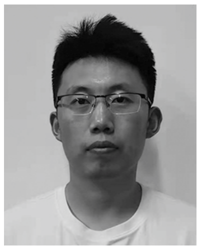
[28] A. Z. da Costa, H. E. H. Figueroa, and J. A. Fracarolli, "Computer vision based detection of external defects on tomatoes using deep learning," *Biosystems Eng.*, vol. 190, pp. 131–144, Feb. 2020.
[29] B. Liu, Z. Ding, L. Tian, D. He, S. Li, and H. Wang, "Grape leaf disease identification using improved deep convolutional neural networks," *Frontiers Plant Sci.*, vol. 11, Jul. 2020.
[30] M. A. Khan, T. Akram, M. Sharif, and T. Saba, "Fruits diseases classification: Exploiting a hierarchical framework for deep features fusion and selection," *Multimedia Tools Appl.*, vol. 79, nos. 35–36, pp. 25763–25783, Sep. 2020.
[31] I. Ahmad, M. Hamid, S. Yousaf, S. T. Shah, and M. O. Ahmad, "Optimizing pretrained convolutional neural networks for tomato leaf disease detection," *Complexity*, vol. 2020, pp. 1–6, Sep. 2020.
[32] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," Oct. 2017, *arXiv:1710.05941*. [Online]. Available: http://arxiv.org/abs/1710.05941
[33] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "DropBlock: A regularization method for convolutional networks," Oct. 2018, *arXiv:1810.12890*. [Online]. Available: http://arxiv.org/abs/1810.12890

FU YUESHENG received the B.S. degree in engineering from Weifang University, in 2017. He is currently pursuing the M.S. degree in engineering with Shandong University of Science and Technology. He has been involved in many projects about non-standard mechanical equipment development, industrial control, and robot control. His current research interests include industrial control and machine vision.

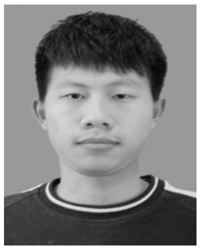
SONG JIAN received the M.S. and Ph.D. degrees in engineering from China Agricultural University, Beijing, China, in 2003 and 2006, respectively. He is currently a Professor with the School of Mechatronics and Vehicle Engineering, Weifang University, Weifang, China. His current research interests include robotics and intelligent agricultural equipment, machine vision and image processing, and machine learning.

XIE FUXIANG received the M.E. and Ph.D. degrees from the School of Engineering, South China Agricultural University, Guangzhou, China, in 2009 and 2012, respectively. He is currently an Associate Professor with the School of Mechatronics and Vehicle Engineering, Weifang University, Weifang, China. His current research interests include intelligent agricultural equipment, mechanism design, and robotic dynamics.

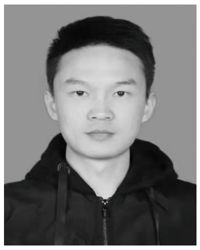
BAI YANG received the B.S. degree in engineering from Northeastern University, in 2012, and the Ph.D. degree from China Agricultural University, in 2019. He is currently working with Weifang University. He has participated in many industrial control projects. His research interests include industrial control and deep learning.



ZHENG XIANG received the B.S. degree in engineering from Zaozhuang College, in 2019. He is currently pursuing the M.S. degree in engineering with Shandong University of Science and Technology. He is involved in several robotics research projects. His current research interest includes machine vision.



GAO PENG received the B.S. degree in engineering from Heze College, in 2019. He is currently pursuing the M.S. degree in engineering with Shandong University of Science and Technology. His current research interests include target detection and machine learning.



WANG ZHENGTAO graduated from Binzhou College, in 2017. He is currently pursuing the M.S. degree in engineering with Shandong University of Science and Technology. He is involved in several robotics research projects. His current research interests include mechanical control and machine vision.

XIE SHENGQIAO received the B.S. degree in engineering from Shandong University of Technology, in 2019. He is currently pursuing the M.S. degree in engineering with China Agricultural University, Beijing, China. He has been involved in several projects on target detection. His research interests include machine vision and image processing.