

Automatic Identification of Tomato Pests Using Parallel Deep Learning Models

Mei-Ling Huang* and You-An Chen

Department of Industrial Engineering & Management, National Chin-Yi University of Technology,
57, Sec. 2, Chung-Shan Rd., Taiping, Taichung, Taiwan

(Received December 20, 2023; accepted May 13, 2024)

Keywords: parallel deep learning model, image classification, tomato pest

Owing to the increasingly serious greenhouse effect and rising global temperatures, pest reproduction and metabolism will accelerate, which will lead to significant reductions in crop yields. To date, many studies have applied deep learning models to pest identification tasks. However, there are many pest types with similar shapes, so in this study, we propose a parallel deep learning model with an attention mechanism module to improve the classification of tomato pest species. We used a public dataset and selected *Bemisia tabaci*, *Helicoverpa armigera*, *Myzus persicae*, *Spodoptera exigua*, *Spodoptera litura*, *Thrips palmi*, *Tetranychus urticae*, and *Zeugodacus cucurbitae*. These eight common tomato pests were selected with a total of 412 original images. The original images were enhanced to 1655 images through horizontal flipping and angle rotation. The proposed ECA-Xception-MobileNet (EXM-Net) extracted image features on the basis of Xception and MobileNetV2, added an Efficient Channel Attention (ECA) attention mechanism before the global average pooling layer, and then used the convolution operation to fuse the two model outputs to enhance model performance. The accuracy, precision, recall, F1-score, and PR-AUC score after data augmentation were 98.72, 98.44, 98.86, 99.41, and 99.76%, respectively. After experiments and testing on different datasets, it was confirmed that EXM-Net performs better than a single model and has a high degree of generalization ability. The proposed EXM-Net uses two deep learning models to extract and fuse different features to make up for important features missed by a single model, and combines the attention mechanism module to improve model efficiency and generalization capabilities.

1. Introduction

Insect pests are a major factor in reducing the economic value of crops. Therefore, strengthening pest monitoring and prevention capabilities is critical for food security and protecting agricultural economies. However, the key to achieving this goal is to identify pests rapidly and accurately so that effective recommendations can be made about where the infestation occurs and what subsequent measures to take. Traditional pest identification relies on experienced practitioners to identify pests on the basis of their external characteristics, which is

*Corresponding author: e-mail: huangml@ncut.edu.tw
<https://doi.org/10.18494/SAM4837>

cumbersome and time-consuming. With the rapid development of artificial intelligence and computer vision technology, researchers have proposed many automatic pest identification methods to alleviate these problems.⁽¹⁾ Ayan *et al.*⁽²⁾ proposed an ensemble convolutional neural network model called GAEnsemble in 2020. The GAEnsemble model integrated the best three models including Inception-V3, Xception, and MobileNet from seven different pretraining models. The results showed that the proposed ensemble model achieved 98.81, 95.16, and 67.13% accuracies in classifying D0, SMALL, and IP102 datasets, respectively. Khanramaki *et al.*⁽³⁾ proposed an ensemble classifier to identify citrus pests in 2021. First, the original RGB image is converted into various spaces to form different feature subsets, and then an integrated classifier composed of AlexNet, VGG16, ResNet 50, and Inception-ResNet-v2 is used for classification. The results showed that the proposed model achieved an accuracy of 99.04% for the multi-classification of citrus plant pest images.

Wang *et al.*⁽⁴⁾ proposed a combination of ConvNeXt and Swin Transformer, leveraging the different advantages of the two models to achieve complementary effects, and inputted the extracted feature maps into the multilayer residual block (MIX-Block) fusion to eliminate duplication of the two models to learn more complex features. The results showed that the proposed architecture achieved 76.1, 93.1, and 98.5% accuracies in classifying IP102, insect, and D0 datasets, respectively. Abade *et al.*⁽⁵⁾ proposed a soybean crop pest dataset called NemaDataset in 2022 and a convolutional neural network architecture called NemaNet. The architecture uses DenseNet121 and InceptionV3 to extract features. The feature maps extracted by the two models are combined and finally input to the fully connected layer for classification. The results showed that the transfer learning architecture achieved an accuracy of 98.82% for pest classification. It can be seen that the deep learning model can effectively identify pest species to help farmers make rapid decisions.

Tomato (*Solanum lycopersicum*) is one of the most important fruit and vegetable crops in the world because of its high nutritional value and it could be processed into various products. According to the Food and Agriculture Organization (FAO) of the United Nations, pests cause significant damage to the world's total crop production every year. Such economic losses can be reduced if pests and diseases affecting tomato crops are identified early, and appropriate tillage and pest control are provided to meet crop growth needs.⁽⁶⁾ Thus far, only a few scholars have conducted research on tomato pest species identification,^(6–8) and most other scholars have conducted research on tomato disease identification.^(9–18) In this study, we took eight common tomato pests as research objects to assist farmers in identifying tomato pests.

Thus far, parallel models have been used in many fields, such as disease detection,^(19,20) plant leaf disease identification,^(21–24) and pest identification.^(25,26) The aforementioned studies have achieved good results in various fields. These parallel models only use original deep models and are not combined with other improvement methods, such as the attention mechanism and knowledge distillation, to achieve breakthroughs in model performance. In addition to proposing a parallel model XM-Net, we combined it with the attention mechanism module to form an EXM-Net with better performance on tomato pest identification. The contributions of this study are as follows:

1. We combined Xception and MobileNetV2 models to propose a parallel XM-Net model.
2. We proposed a parallel EXM-Net model to optimize the parallel model structure by adding an attention mechanism module.
3. The EXM-Net model exhibits both the advantages of the parallel model and attention mechanism module.

Figure 1 shows the research steps of this study. First, the dataset is split into training, validation, and test sets at a ratio of 60:20:20. The images in the training and validation sets are augmented through rotations and flips to expand the number of images. The performance characteristics of VGG16, ResNet50, EfficientNetB0, Xception, MobileNetV2, and InceptionV3 are compared to select the top two models to form the proposed parallel XM-Net model. Finally, the proposed EXM-Net model adds an attention module to the XM-Net model to enhance model performance.

2. Materials and Methods

2.1 Tomato pest dataset

The dataset used in this study was selected from Ref. 6. Eight common tomato pests, namely, *Bemisia tabaci* (BA), *Helicoverpa armigera* (HA), *Myzus persicae* (MP), *Spodoptera exigua* (SE), *Spodoptera litura* (SL), *Thrips palmi* (TP), *Tetranychus urticae* (TU), and *Zeugodacus cucurbitae* (ZC), were selected with a total of 412 original images. Table 1 shows the number of original images for training, validation, and test groups in each class. We applied horizontal flipping and several rotation angles to expand the number of images, increasing the original 412

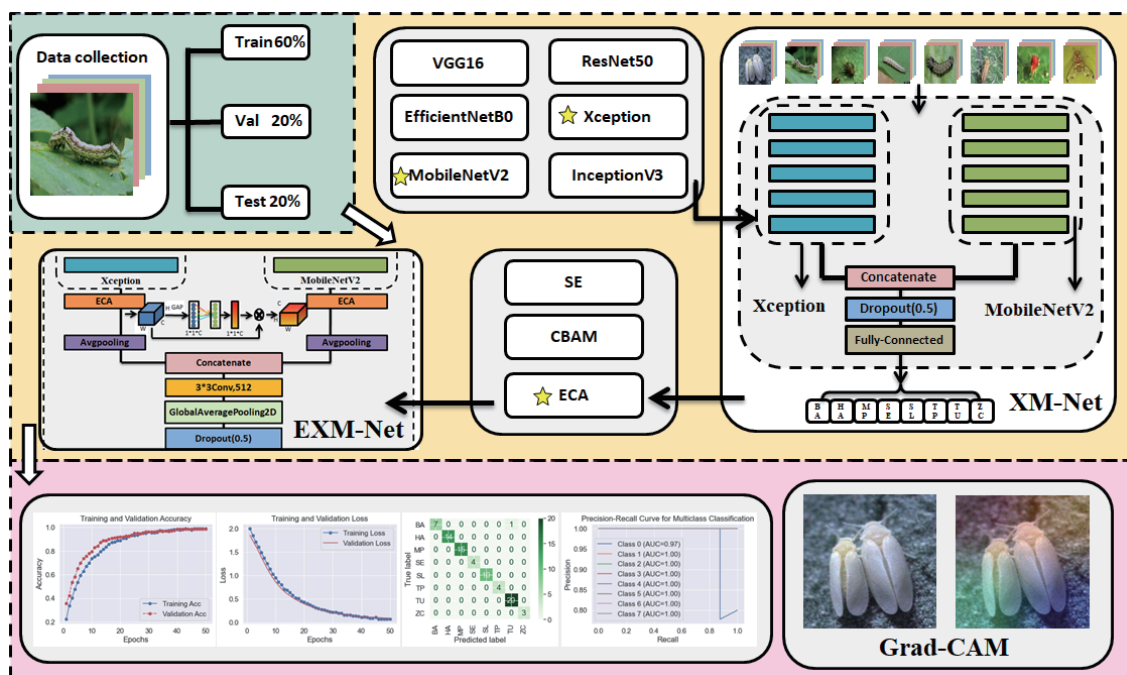


Fig. 1. (Color online) Study flowchart.

Table 1
Number of original images in each class.

Class name	Number of original images			Total
	Train	Validation	Test	
BA	28	7	8	43
HA	48	11	14	73
MP	49	12	15	76
SE	9	3	4	16
SL	35	8	10	53
TP	21	5	4	30
TU	67	16	20	103
ZC	12	3	3	18
Total				412

Table 2
Number of images in each class after augmentation.

Class name	Number of augmentation images			Total
	Train	Validation	Test	
BA	132	32	8	172
HA	224	55	14	293
MP	235	58	15	308
SE	47	11	4	62
SL	162	40	10	212
TP	89	22	4	115
TU	316	78	20	414
ZC	58	18	3	79
Total				1655

images to 1655 images. Table 2 shows the number of images for training, validation, and test groups in each class after augmentation. Figure 2 shows examples of tomato pests, and Fig. 3 shows examples of data augmentation.

2.2 Attention networks

The attention mechanism selects important features on the basis of different weight scores to improve model performance. Initially, the attention mechanism was applied in natural language processing, and it has now been extended to image, speech, and other data processing tasks. There are three common attention mechanisms in the field of deep learning, namely, squeeze-and-excitation network (SENet), efficient channel attention network (ECANet), and convolutional block attention module (CBAM).

2.2.1 SENet

SENet was proposed by the autonomous driving company Momenta in 2017 and won the championship in the annual ILSVR classification competition. This model improves the model

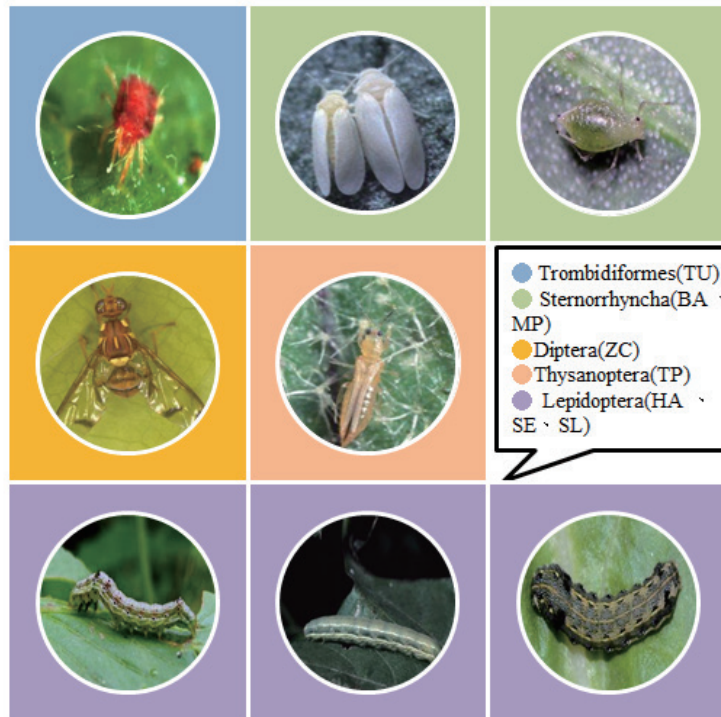


Fig. 2. (Color online) Examples of tomato pests.

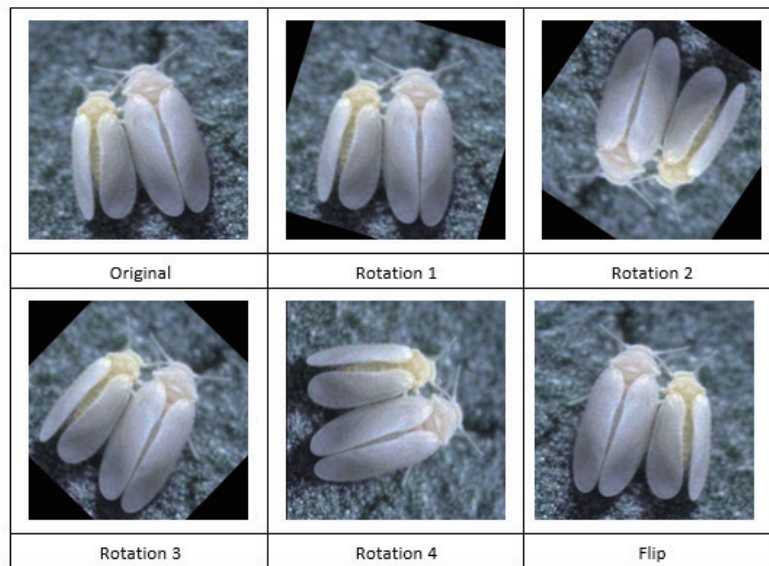


Fig. 3. (Color online) Examples of data augmentation.

accuracy by selecting important features. Because the SE module is easy to implement, it is often applied in various research projects. As illustrated in Fig. 4, the SE module is mainly divided into three parts: squeeze, excitation, and reweight. The details of these parts are explained below.⁽²⁷⁾

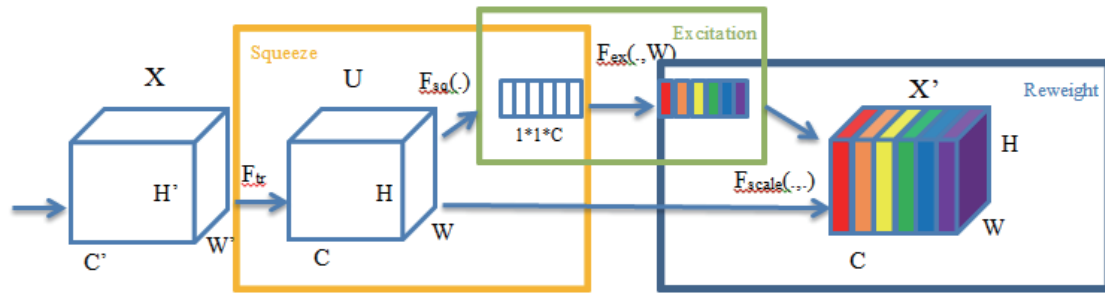


Fig. 4. (Color online) Architecture of SENet.

1. Squeeze: Perform global average pooling on the three axes $C \times H \times W$ of the input image to turn all information of points in space into mean values.
2. Excitation: Use two fully connected layers to reduce and then increase the dimensionality of the squeeze output results, and learn the weight coefficients of each channel.
3. Reweight: Multiply the weight of the excitation output by the previous feature to complete the weight update of the original feature channel.

2.2.2 ECANet

An ECANet utilizes an ECA module to improve model performance. First, the ECA module performs global average pooling on the input feature map, and then uses a 1D convolution kernel to exchange information between local channels, so that the channel dimension is proportional to the size of the convolution kernel, thereby ensuring model calculation efficiency and performance. Figure 5 shows the architecture of ECANets.⁽²⁸⁾

2.2.3 CBAM

The CBAM combines channel and spatial attention mechanisms and has been used in many common convolutional neural networks. It has been confirmed that it can effectively improve the performance of convolutional neural networks for tasks such as image classification and object detection. As shown in Fig. 6, the CBAM is mainly divided into a channel attention module and a spatial attention module. The details of the two modules are explained below.⁽²⁹⁾

1. Channel attention module: First, the feature map is input to the global maximum and average pooling layers, and then processed by the shared multilayer perceptron multilayer perceptron. The two results are added and passed through the sigmoid activation function to generate the weight of each channel, and finally, we multiply the weight of each channel with the input feature map.
2. Spatial attention module: We extract the maximum and average values of each feature point in the feature map output by the channel attention mechanism for concatenation. Then, we use a convolution layer with one channel to reduce the dimension and pass the sigmoid activation function to generate a spatial attention feature map. Finally, the feature map is multiplied by the input feature map to obtain the final feature map.

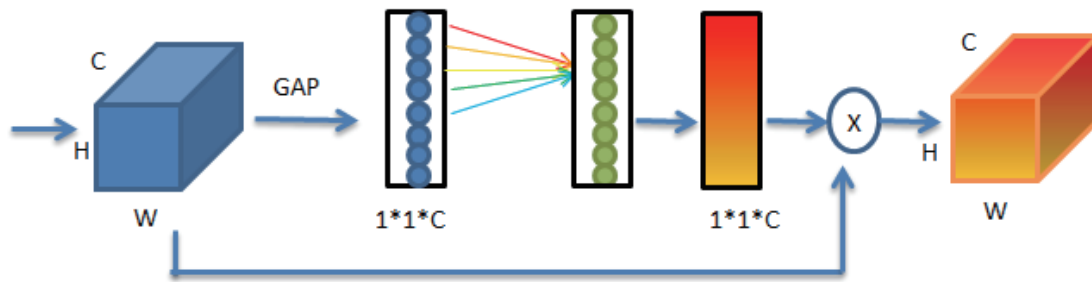


Fig. 5. (Color online) Architecture of ECANets.

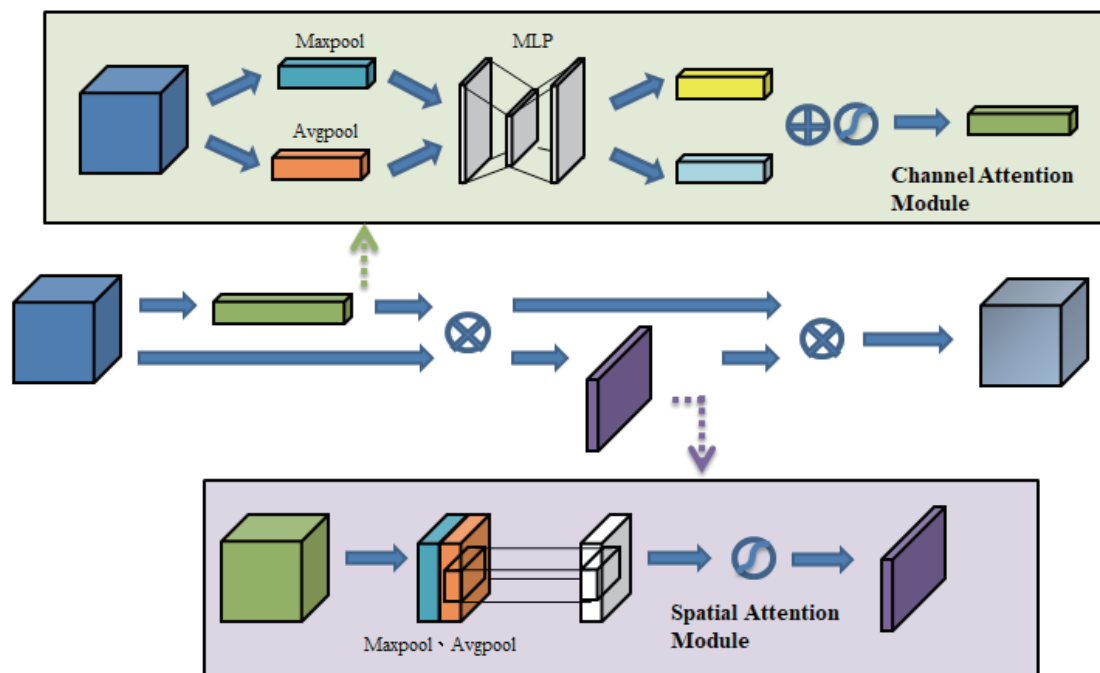


Fig. 6. (Color online) CBAM module.

2.3 Architecture of networks

2.3.1 VGGNet

VGGNet is a deep convolutional neural network jointly developed by the Oxford University Computer Vision Research Group and DeepMind. The network not only has good performance, but also has strong scalability and excellent generalization.⁽³⁰⁾ The network structure is simple and easy to implement, including VGG16 with 13 convolutional layers, three fully connected layers, and one output layer. VGG19, with three additional convolutional layers, adds a maximum pooling layer after the convolutional layer and ReLU activation function to avoid the vanishing gradient problem. VGGNet replaces larger convolution kernels by stacking multiple smaller convolution kernels to maintain the size of the receptive field. It can also reduce the number of parameters while increasing the number of nonlinear mapping.⁽³⁰⁾ Figure 7 shows the architecture of VGG16.

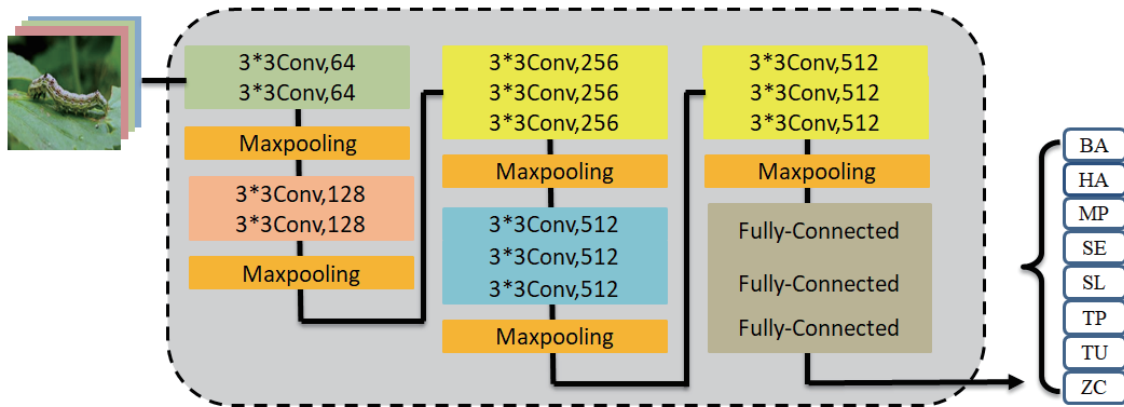


Fig. 7. (Color online) Architecture of VGG16.

2.3.2 ResNet

The success of ResNet lies in the use of the residual network to solve the problem of performance degradation as the network depth increases.⁽³¹⁾ Residual modules are mostly composed of two or three convolution modules, identity mapping, and skip connections. By using skip connections, information can flow directly from shallow layers to deep layers to alleviate the vanishing gradient problem. The ResNet model architecture is based on VGG19 and is modified and added with a residual module. The model has been pretrained in the ImageNet large-scale image dataset, so it is often used for tasks such as image classification and target detection in deep learning.⁽³¹⁾ Figure 8 shows the architecture of ResNet50.

2.3.3 EfficientNet

Google proposed a new composite network scaling method in 2019. For high-resolution images, the deeper the network, the better the reception field, and the wider the network, the more detailed the features. EfficientNetB0 draws on MnasNet for a multi-objective neural structure search and uses the same MBConv as MobileNetV2 as the backbone while optimizing the model performance and floating point operations (FLOPS).⁽³²⁾ Figure 9 shows the architecture of EfficientNetB0.

2.3.4 Xception

On the basis of the InceptionV3 model, Xception uses the depth-wise separable concatenate to replace the original Inception module to separate the space and channels, which reduces the network complexity, improves the model performance, and reduces the model parameters. The same as InceptionV3, Xception consists of Entry, Middle, and Exit modules. Similar to the ResNet model, the residual modules enable Xception fast convergence to reduce the training time.⁽³³⁾ The architecture of Xception is shown in Figure 10.

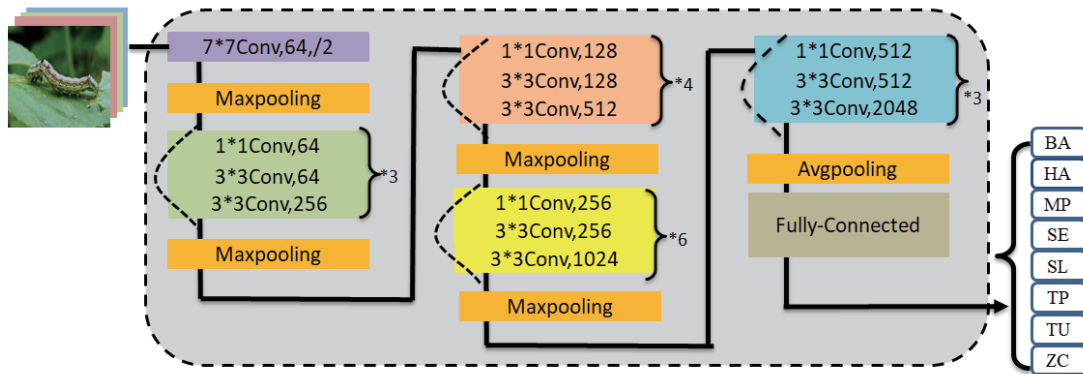


Fig. 8. (Color online) Architecture of ResNet50.

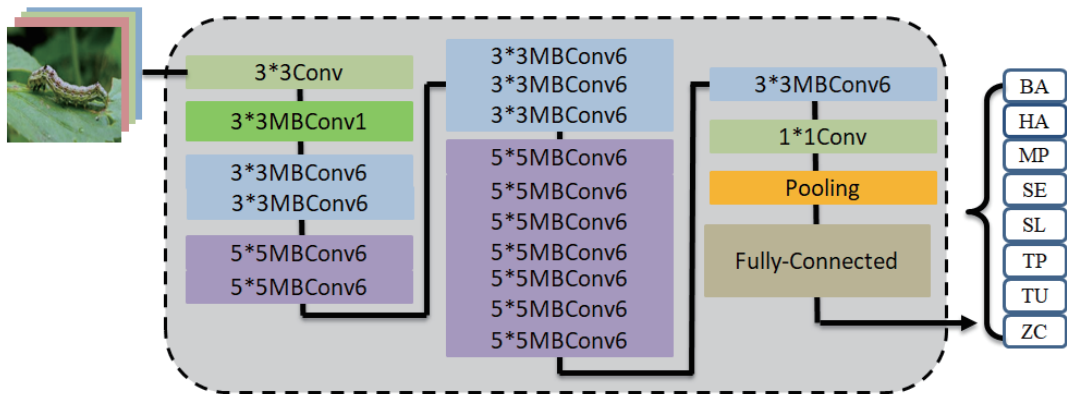


Fig. 9. (Color online) Architecture of EfficientNetB0.

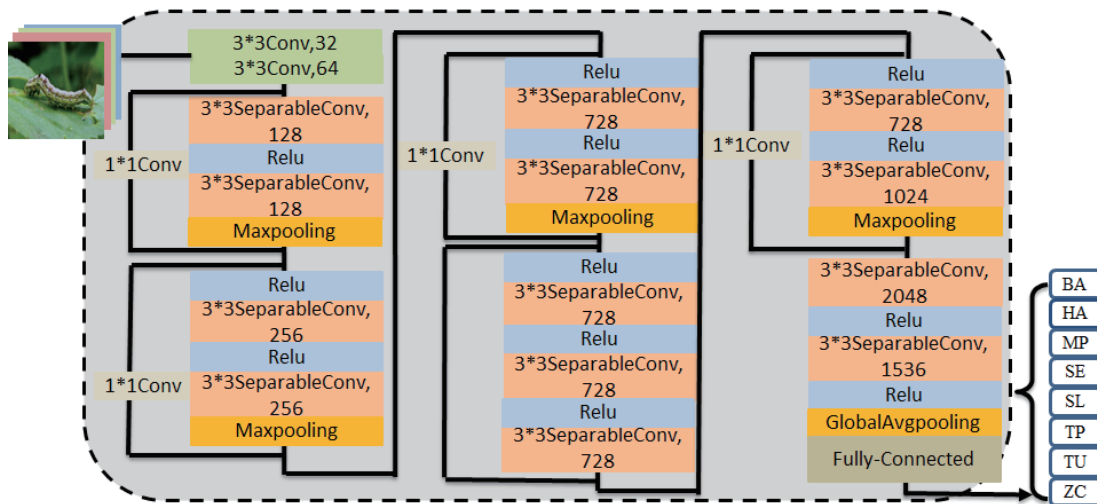


Fig. 10. (Color online) Architecture of Xception.

2.3.5 MobileNetV2

On the basis of MobileNetV1, MobileNetV2 adds a residual network to reduce model calculation costs, accelerate model convergence, and maintain feature extraction capabilities. MobileNetV2 uses a linear bottleneck in the residual network to ensure that the model has higher feature transfer and learning capabilities. MobileNetV2 is composed of three convolution layers, seven bottleneck residual blocks, and one average pooling layer.⁽³⁴⁾ Figure 11 shows the architecture of MobileNetV2.

2.3.6 InceptionV3

The main improvement of InceptionV3 is the use of smaller and asymmetrical convolutional layers to increase the width of the model architecture to generate high-dimensional feature maps. Compared with GoogleNet, InceptionV3 uses only one auxiliary classifier at the end of the model to achieve a function similar to dropout regularization, helping the model to be more efficient and stable. InceptionV3 consists of the Inception module, convolution layers, and maxpooling layer.⁽³⁵⁾ Figure 12 shows the architecture of InceptionV3.

2.4 Hyperparameter optimization

In this study, we used a hyperparameter optimization method called Tree-structured Parzen Estimator⁽³⁶⁾ to study the optimal parameter combination. This method is improved on the basis of the Bayesian optimization method by constructing two Gaussian mixture models to simulate the probabilities of good and bad results, and to evaluate the quality of the combination. We repeat this process until the optimal hyperparameter combination or a certain number of iterations is found. Several related research results have confirmed that this algorithm can find better hyperparameter combinations in fewer evaluations, making it computationally efficient to

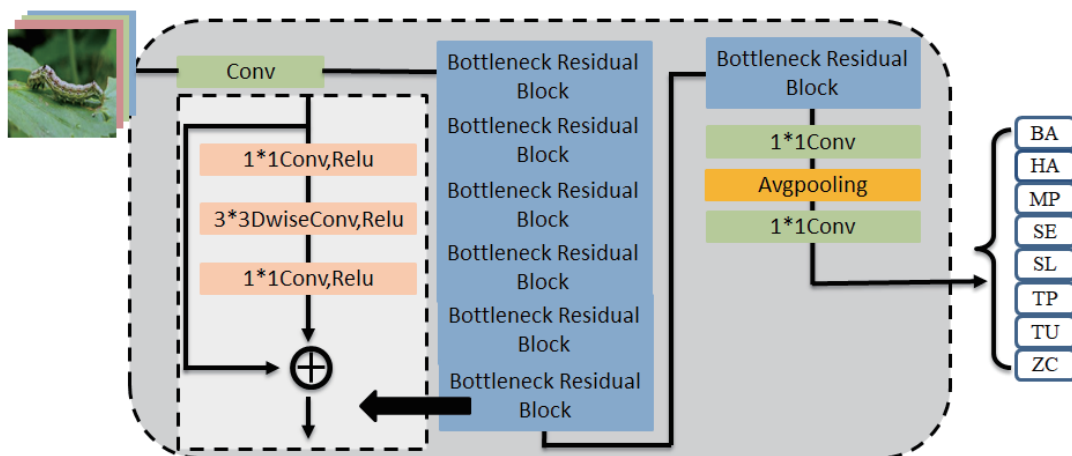


Fig. 11. (Color online) Architecture of MobileNetV2.

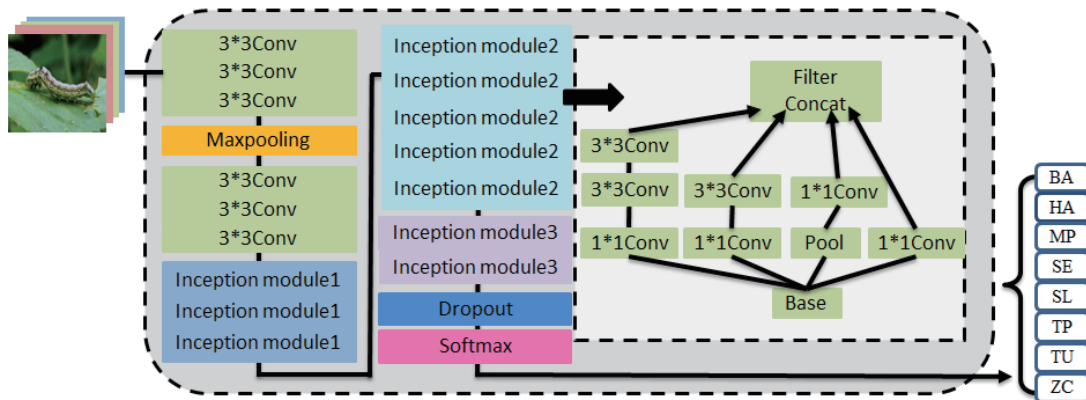


Fig. 12. (Color online) Architecture of InceptionV3.

perform optimization tasks. Owing to the limitation of the memory capacity of the hardware device, we only consider batch sizes from 4 to 8, epochs from 30 to 50, and learning rates from 0.01 to 0.00001.

2.5 Gradient-weighted class activation mapping (Grad-CAM)

In this study, we used gradient-weighted class activation mapping (Grad-CAM) to visualize the feature map of the convolutional neural network. Grad-CAM performs backpropagation on the final feature map of the model, calculates the gradient information of the same size as the feature map, and sums up the weights of different channels as the result to obtain a heat map of the concerned area.

2.6 Computer equipment

The equipment used in the experiment is an Intel® Core™ i7-10700 2.90 GHz CPU with 32 GB of memory and an NVIDIA GeForce RTX 2070 GPU. The whole experiment process is performed using Python 3.8 (Python Software Foundation, Fredericksburg, Virginia, USA), which contains Keras 2.6 and Tensorflow GPU 2.6.0.

3. Results

3.1 Model performance characteristics

The model performance characteristics before data augmentation are shown in Table 3. Except InceptionV3, the accuracies from the other models are above 0.70. The top two accuracies are 0.8205 and 0.7948, which originate from Xception and MobileNetV2, respectively. The model performance characteristics after data augmentation are shown in Table 4. Compared with the results in Table 3, all the performance characteristics are improved significantly after data augmentation. The same as the finding in Table 3, except InceptionV3, the accuracies from

Table 3
Model performance characteristics before data augmentation.

Model	Accuracy	Recall	F1-score	Precision	PR-AUC
VGG16	0.7051	0.5988	0.5482	0.5104	0.6943
ResNet50	0.7692	0.6873	0.6737	0.6811	0.8145
EfficientNetB0	0.7435	0.6383	0.6299	0.6392	0.7265
Xception	0.8205	0.7108	0.7191	0.7491	0.7956
MobileNetV2	0.7948	0.7738	0.7656	0.7694	0.7528
InceptionV3	0.5000	0.3942	0.3920	0.4576	0.4079

Table 4
Model performance characteristics after data augmentation.

Model	Accuracy	Recall	F1-score	Precision	PR-AUC
VGG16	0.9487	0.8906	0.9021	0.9457	0.9379
ResNet50	0.9487	0.9343	0.9328	0.9331	0.9384
EfficientNetB0	0.9359	0.9410	0.9250	0.9174	0.9698
Xception	0.9615	0.9562	0.9613	0.9683	0.9535
MobileNetV2	0.9615	0.9406	0.9528	0.9706	0.9535
InceptionV3	0.8333	0.7836	0.7824	0.8123	0.8513

the other models are above 0.90 in Table 4. The top two accuracies are 0.9615 and 0.9615, which originate from Xception and MobileNetV2, respectively. The above two models were selected to form the proposed parallel XM-Net model (see Sec. 2.3.7).

3.2 Proposed parallel XM-Net

The performance characteristics of VGG16, ResNet50, EfficientNetB0, Xception, MobileNetV2, and InceptionV3 were compared to select the top two models (Xception and MobileNetV2) to form the proposed parallel XM-Net model. Different from the ensemble model training basic models individually, the parallel model concatenates feature maps outputted from basic models to increase the execution speed. Figure 13 shows the design of the proposed parallel XM-Net model.

3.3 Adding attention mechanisms

Three common attention mechanisms, namely, SE, ECA, and CBAM, were combined with the proposed parallel XM-Net model. The attention mechanisms were added before the average pooling layer for both Xception and MobileNetV2 to form SE-XM-Net, ECA-XM-Net, and CBAM-XM-Net. Among these three models, the model with highest performance became the final proposed EXM-Net model.

Results of the proposed parallel XM-Net model and the three common attention mechanisms, namely, SE, ECA, and CBAM, added to the proposed parallel XM-Net model are shown in Table 5. The accuracies from the proposed parallel XM-Net, SE+XM-Net, CBAM+XM-Net, and

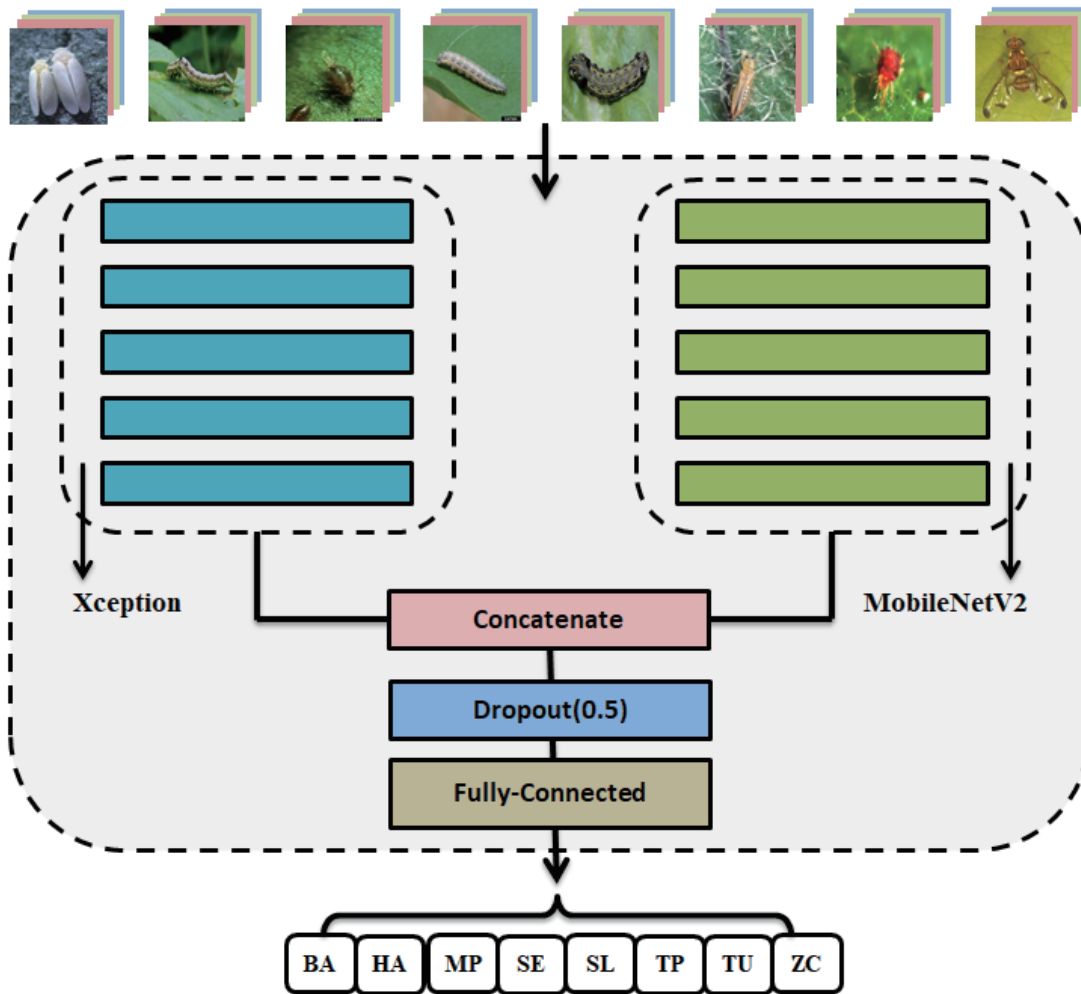


Fig. 13. (Color online) Architecture of the proposed parallel XM-Net.

Table 5
Model performance characteristics after adding attention mechanism.

Attention	Accuracy	Recall	F1-score	Precision	PR-AUC
XM-Net	0.9743	0.9687	0.9750	0.9862	0.9960
SE+XM-Net	0.8974	0.8223	0.7607	0.7381	0.9890
CBAM+XM-Net	0.9743	0.9687	0.9743	0.9852	0.9960
ECA+XM-Net	0.9872	0.9844	0.9876	0.9922	0.9967

ECA+XM-Net models are 0.9743, 0.8974, 0.9743, and 0.9872, respectively. Adding the SE module degrades the performance of the proposed parallel XM-Net model, whereas there is no significant difference in the effect of the model with or without adding the CBAM module. Adding the ECA attention mechanism to the proposed parallel XM-Net model achieves the highest performance, and finally, the combination becomes the proposed EXM-Net model in this study. The performance plots after adding the attention mechanism for the proposed EXM-

Net model are shown in Fig. 14; Fig. 15 shows the architecture of the proposed parallel EXM-Net model.

3.4 Ablation experiments

In this section, we compare the model performance by adding Dropout and the attention mechanism ECA using images after augmentation. The accuracy was increased from 0.9487 to 0.9615 for Xception by adding Dropout. The accuracy for the parallel model Xception+ MobileNetV2 reached 0.9872 when Dropout and ECA were used (Experiment 4 in Table 6). As shown in Table 6, the accuracy, loss, confusion matrix, and PR-AUC from experiments 3 and 4 are clearly higher than those from experiments 1 and 2. Adding Dropout prevents overfitting problems at the end of Xception, thereby improving the model performance. After the two models are paralleled, the model learns more detailed features by complementing each other's

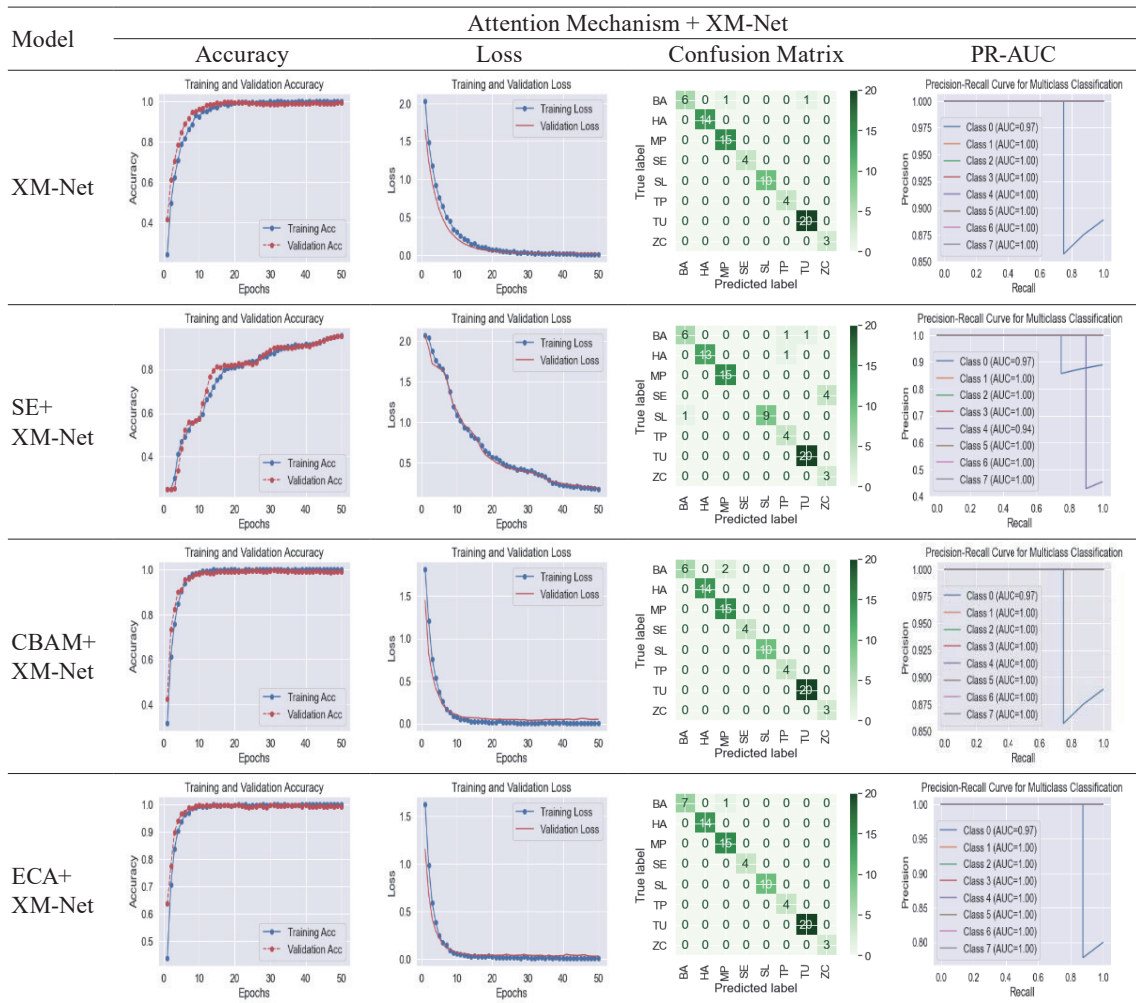


Fig. 14. (Color online) Performance plots after adding attention mechanism.

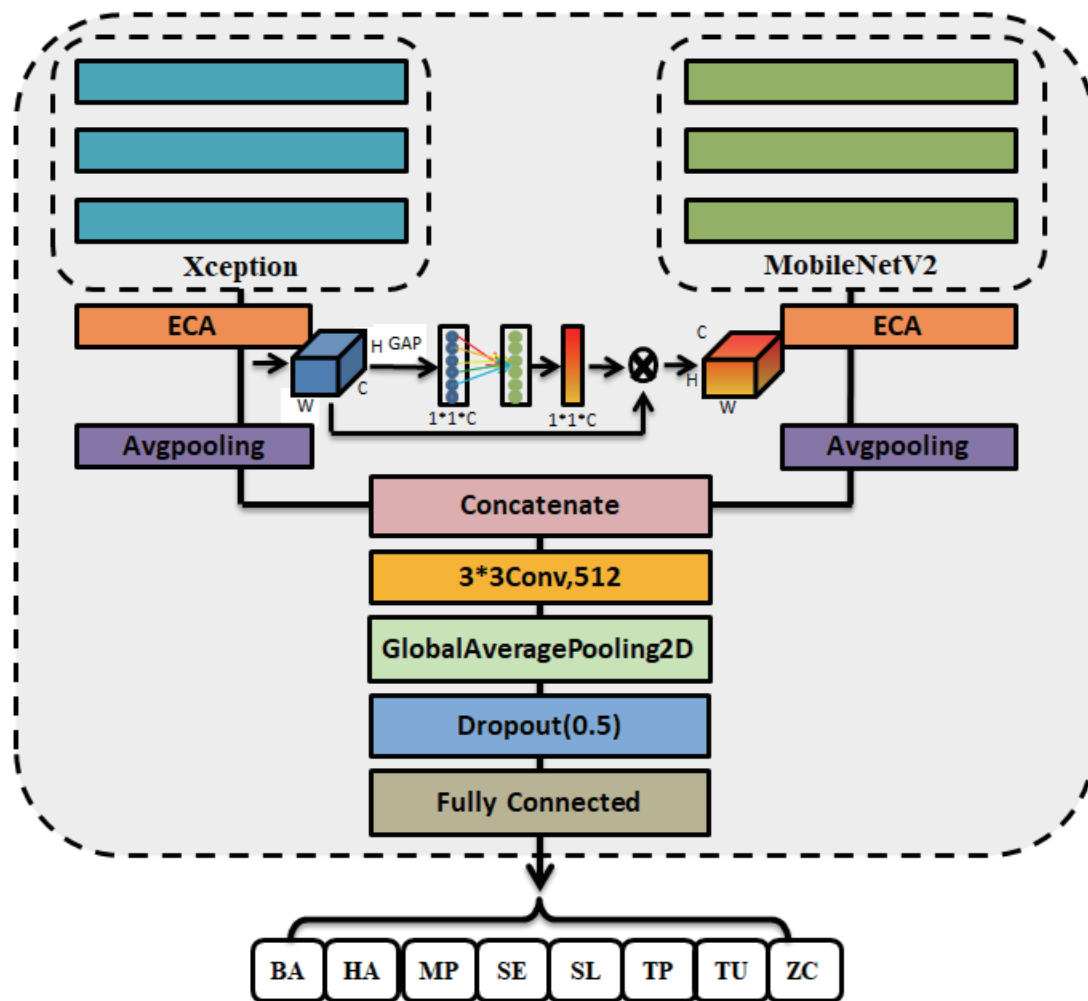


Fig. 15. (Color online) Architecture of EXM-Net.

Table 6
Results of ablation experiments.

	Xception	MobileNetV2	Dropout	ECA	Accuracy	Recall	F1-score	Precision	PR-AUC
(1)	⊙	—	—	—	0.9487	0.9250	0.9391	0.99600	0.9379
(2)	⊙	—	⊙	—	0.9615	0.9562	0.9613	0.9683	0.9535
(3)	⊙	⊙	⊙	—	0.9743	0.9687	0.9750	0.9862	0.9960
(4)	⊙	⊙	⊙	⊙	0.9872	0.9844	0.9876	0.9922	0.9967

missing features, and the accuracy is 0.0385 higher than that of a single model. It can be seen from Table 6 that the parallel model adding the ECA attention module achieves the highest accuracy of 0.9872.

3.5 Results of hyperparameters

The optimal combination for the dataset before data augmentation after five cross-validations was *batch size* = 4, *epochs* = 50, and *learning rate* = 1.3777832919010947e−05, which results in

the higher accuracy, recall, F1-score, precision, and PR-AUC of 0.8461, 0.8290, 0.8362, 0.8571, and 0.8964, respectively. For the augmented dataset, the optimal combination for the dataset before data augmentation after five cross-validations was *batch size* = 8, *epochs* = 50, and *learning rate* = $1.5151670794642941e-06$, but there were only slight improvements in accuracy and F1-score. Table 7 shows the results of the models with hyperparameters.

3.6 Results of feature visualization

It can be seen from the visualization results in Fig. 16 that the focus of EXM-Net is more comprehensive than that of a single model. The two-branch model provides rich pest features instead of redundant features in the background, which improves EXM-Net for identifying tomato pests while reducing the attention of complex backgrounds.

4. Discussion

4.1 Comparison with SOTA

Table 8 presents the performance characteristics from related studies. Sun *et al.*⁽³⁸⁾ improved the SE attention mechanism according to the architecture of the fire module in SqueezeNet by changing the convolution kernel size to adjust the extracted features and formed S1 and S2 modules with different sizes. The two are joined to SqueezeNet to form SSNet. After experiments on module placement and quantity, the results showed that SSNet achieved a good accuracy of 98.06%. However, the authors did not consider the impact of other attention mechanisms on SqueezeNet. Chen *et al.*⁽³⁹⁾ proposed the feature positioning module EFLM and the adaptive filtering fusion module AFFM to improve pest identification capabilities. The results showed that the accuracy of the network with ResNet50 as the backbone reached 100%. Although the network performance is amazing, the architecture requires high computer power, which is not conducive to easy application by farmers. Huang *et al.*⁽⁶⁾ constructed an architecture ResNet50+DA that combines a deep learning model and a machine learning classifier, and used Bayesian optimization methods to find the optimal hyperparameter combination. The results showed that the architecture achieved an accuracy of 0.9712. However, in this study, the authors only used a self-created tomato pest dataset and did not use other datasets to confirm that the proposed architecture can be widely used in different fields.

Table 7
Results of hyperparameters.

Model	Dataset	Accuracy	Recall	F1-score	Precision	PR-AUC
EXM-Net	Original	0.8461	0.7681	0.7642	0.7692	0.8461
EXM-Net (Hyperparameter)	Original	0.8461	0.8290	0.8362	0.8571	0.8964
EXM-Net	Augmented	0.9872	0.9844	0.9876	0.9922	0.9967
EXM-Net (Hyperparameter)	Augmented	0.9872	0.9844	0.9886	0.9941	0.9967






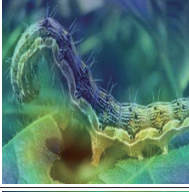
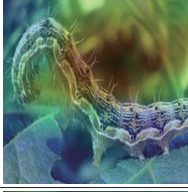


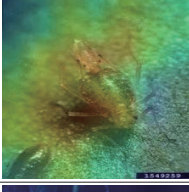
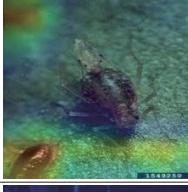
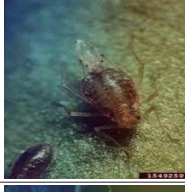
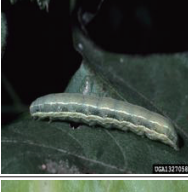

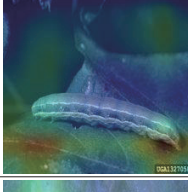
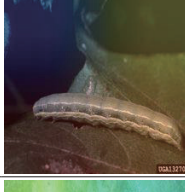

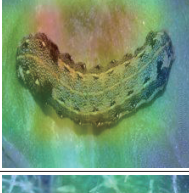
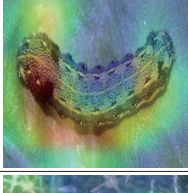
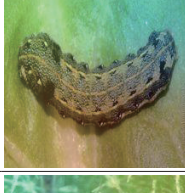




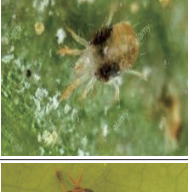
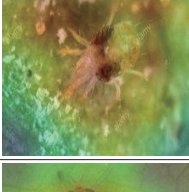
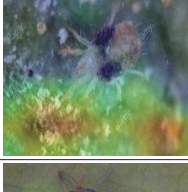



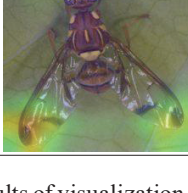
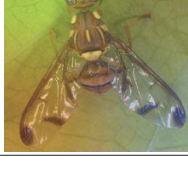
Class	Original	MobileNetV2	Xception	EXM-Net
BA				
HA				
MP				
SE				
SL				
TP				
TU				
ZC				

Fig. 16. (Color online) Results of visualization.

Table 8
Comparison with SOTA.

Literature	Database	Method	Accuracy	Recall	F1-score	Precision
Yu <i>et al.</i> ⁽⁷⁾	Internet ⁽⁷⁾	IResNet50	0.945	0.942	0.943	0.943
Pattnaik <i>et al.</i> ⁽³⁷⁾	Internet ⁽¹⁰⁾	DenseNet169	0.8883	—	—	—
Sun <i>et al.</i> ⁽³⁸⁾	A database of eight common tomato pest images ⁽⁸⁾	SSNet	0.9806	—	—	—
Chen <i>et al.</i> ⁽³⁹⁾	A database of eight common tomato pest images ⁽⁸⁾	Improving branch ResNet50	1.0000	—	—	—
Jia <i>et al.</i> ⁽⁴⁰⁾	Internet ⁽⁵⁾	Inception-v3	0.869	—	—	—
Huang <i>et al.</i> ⁽⁶⁾	IPM, NBAIR, Google ⁽⁸⁾	ResNet50+DA VGG16	0.9712 0.9495	0.9745 0.9466	0.9608 0.9495	0.9525 0.9567
Proposed Method	IPM, NBAIR, Google ⁽⁸⁾	XM-Net	0.9743	0.9687	0.9750	0.9862
		EXM-Net	0.9872	0.9844	0.9876	0.9822

Table 9
Performance of related studies on PlantVillage.

Literature	Method	Accuracy	Recall	F1-score	Precision
Amin <i>et al.</i> ⁽⁴¹⁾	ResNet152	0.9837	—	—	—
	InceptionV3	0.9626			
	EfficientNetB0	0.9791			
	DenseNet121	0.9782			
	EfficientNetB0 + DenseNet121	0.9856			
Agarwal <i>et al.</i> ⁽⁴²⁾	Proposed Model	0.94	0.8065	0.9583	—
Waheed <i>et al.</i> ⁽⁴³⁾	Optimized DenseNet	0.9806	0.98	0.98	0.98
Jasrotia <i>et al.</i> ⁽⁴⁴⁾	CLAHE-HSV-CNN	0.9676	0.97	0.97	0.97
Li <i>et al.</i> ⁽⁴⁵⁾	LMBRNet	0.9891	0.9763	0.9762	0.9763
Proposed model	XM-Net	0.9856	0.9846	0.9846	0.9848
	EXM-Net	0.9799	0.9784	0.9783	0.9785

4.2 Additional application of proposed parallel model to PlantVillage dataset

The proposed XM-Net and EXM-Net models were applied to the corn dataset from PlantVillage. From Table 9, the XM-Net model achieved the accuracy, recall, F1-score, and precision of 0.9856, 0.9846, 0.9846, and 0.9848, whereas those of the EXM-Net model were 0.9799, 0.9786, 0.9783, and 0.9785, respectively. As compared with the results from related studies using the same dataset, the accuracy from Li *et al.*⁽⁴⁵⁾ reached 0.9891, which is 0.0035 higher than that from our proposed XM-Net model. The results from Ref. 45 could benefit from applying a 1×1 -size depth-separable convolution layer to improve the model performance. However, the two parallel models proposed in this study have higher recall, F1-score, and precision values than those shown in Ref. 45.

5. Conclusion

In this study, we proposed two parallel models, XM-Net and EXM-Net, to identify common tomato pests. XM-Net parallels two models, Xception and MobileNetV2, to extract different features. In addition, EXM-Net adds an ECA attention mechanism module to XM-Net to focus on more delicate features to further improve the model performance. The results showed that the accuracy, precision, recall, F1-score, and PR-AUC score after data augmentation from the proposed EXM-Net model reached 0.9872, 0.9844, 0.9886, 0.9941, and 0.9967, respectively.

Currently, we only selected eight common types of tomato pest, and other pests still need to be collected and identified for a more comprehensive pest control in a future study. In addition, the proposed XM-Net and EXM-Net models are only designed to achieve high accuracy, and the model parameters and model size are not considered in the experiment. In the future, a lightweight model is encouraged to be matched with mobile devices to construct a real-time and efficient pest identification system.

Compliance with Ethical Standards

Conflict of Interest: All the authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

Acknowledgments

The authors gratefully acknowledge the financial support of the Ministry of Science and Technology of Taiwan, R.O.C., through grant no. MOST 111-2221-E-167-007-MY3.

CRedit authorship contribution statement

Mei-Ling Huang: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Validation, Visualization, Writing - original draft, Writing - review & editing, Supervision.

You-An Chen: Methodology, Data curation, Software, Formal analysis, Investigation, Writing - original draft.

Declaration of generative AI and AI-assisted technologies during the writing process

During the preparation of this work, the authors did not use any generative AI and AI-assisted technologies during the writing process.

References

- 1 Y. Peng and Y. Wang: Ecol. Inform. **72** (2022) 101846. <https://doi.org/10.1016/J.ECOINF.2022.101846>
- 2 E. Ayan, H. Erbay, and F. Varçın: Comput. Electron. Agric. **179** (2020) 105809. <https://doi.org/10.1016/J.COMPAG.2020.105809>

- 3 M. Khanramaki, E. Askari Asli-Ardeh, and E. Kozegar: *Comput. Electron. Agric.* **186** (2021) 106192. <https://doi.org/10.1016/J.COMPAG.2021.106192>
- 4 C. Wang, J. Zhang, J. He, W. Luo, X. Yuan, and L. Gu: *Eng. Appl. Artif. Intell.* **124** (2023) 106563. <https://doi.org/10.1016/J.ENGAPPAI.2023.106563>
- 5 A. Abade, L. F. Porto, P. A. Ferreira, and F. deBarros Vidal: *Biosyst. Eng.* **213** (2022) 39. <https://doi.org/10.1016/J.BIOSYSTEMSENG.2021.11.016>
- 6 M. L.Huang, T. C. Chuang, and Y. C. Liao: *Sustainable Comput. Inf. Syst.* **33** (2022) 100646. <https://doi.org/10.1016/J.SUSCOM.2021.100646>
- 7 H. Yu, J. Liu, C. Chen, A. A. Heidari, Q. Zhang, and H. Chen: *Comput. Electron. Agric.* **195** (2022) 106805. <https://doi.org/10.1016/J.COMPAG.2022.106805>
- 8 S. R. Rupanagudi, B. S. Ranjani, P. Nagaraj, V. G. Bhat, and G. Thippeswamy: *Proc. 2015 Int. Conf. Commun. Inf. Comput. Technol. (ICCICT 2015)*. <https://doi.org/10.1109/ICCICT.2015.7045722>
- 9 T. Anandhkrishnan and S. M. Jaisakthi: *Sustainable Chem. Pharm.* **30** (2022) 100793. <https://doi.org/10.1016/J.SCP.2022.100793>
- 10 P. Baser, J. R. Saini, and K. Kotecha: *Procedia Comput. Sci.* **218** (2023) 1825. <https://doi.org/10.1016/J.PROCS.2023.01.160>
- 11 R. Karthik, M. Hariharan, S. Anand, P. Mathikshara, A. Johnson, and R. Menaka: *Appl. Soft Comput.* **86** (2020) 105933. <https://doi.org/10.1016/J.ASOC.2019.105933>
- 12 M. Agarwal, S. K. Gupta, and K. K. Biswas: *Sustainable Comput. Inf. Syst.* **28** (2020) 100407. <https://doi.org/10.1016/J.SUSCOM.2020.100407>
- 13 V. Gonzalez-Huitron, J. A. León-Borges, A. E. Rodriguez-Mata, L. E. Amabilis-Sosa, B. Ramírez-Pereda, and H. Rodriguez: *Comput. Electron. Agric.* **181** (2021) 105951. <https://doi.org/10.1016/J.COMPAG.2020.105951>
- 14 T. Sanida, A. Sideris, M. V. Sanida, and M. Dasygenis: *Smart Agric. Technol.* **5** (2023) 100275. <https://doi.org/10.1016/J.ATECH.2023.100275>
- 15 M. Astani, M. Hasheminejad, and M. Vaghefi: *Comput. Electron. Agric.* **198** (2022) 107054. <https://doi.org/10.1016/J.COMPAG.2022.107054>
- 16 A. Abbas, S. Jain, M. Gour, and S. Vankudothu: *Comput. Electron. Agric.* **187** (2021) 106279. <https://doi.org/10.1016/J.COMPAG.2021.106279>
- 17 M. Li, G. Zhou, A. Chen, L. Li, and Y. Hu: *Eng. Appl. Artif. Intell.* **123** (2023) 106195. <https://doi.org/10.1016/J.ENGAPPAI.2023.106195>
- 18 Y. Zhang, S. Huang, G. Zhou, Y. Hu, and L. Li: *Comput. Electron. Agric.* **205** (2023) 107605. <https://doi.org/10.1016/J.COMPAG.2022.107605>
- 19 A. Kazemi, M. E. Shiri, A. Sheikhhahmadi, and M. khodamoradi: *Comput. Biol. Med.* **148** (2022) 105775. <https://doi.org/10.1016/J.COMPBIOMED.2022.105775>
- 20 B. Chen, J. Li, X. Guo, and G. Lu: *Biomed. Signal Process. Control* **53** (2019) 101554. <https://doi.org/10.1016/J.BSPC.2019.04.031>
- 21 B. M. Abuhayi and A. A. Mossa: *Inf. Med. Unlocked* **39** (2023) 101245. <https://doi.org/10.1016/J.IMU.2023.101245>
- 22 V. M. Araújo, A. S. Britto, L. S. Oliveira, and A. L. Koerich: *Neurocomputing* **467** (2022) 427. <https://doi.org/10.1016/J.NEUCOM.2021.10.015>
- 23 G. Hu and M. Fang: *Sustainable Comput. Inf. Syst.* **35** (2022) 100696. <https://doi.org/10.1016/j.suscom.2022.100696>
- 24 M. Ji, L. Zhang, and Q. Wu: *Inf. Process. Agric.* **7** (2020) 418. <https://doi.org/10.1016/J.INPA.2019.10.003>
- 25 A. Abade, L. F. Porto, P. A. Ferreira, and F. deBarros Vidal: *Biosyst. Eng.* **213** (2022) 39. <https://doi.org/10.1016/J.BIOSYSTEMSENG.2021.11.016>
- 26 C. Wang, J. Zhang, J. He, W. Luo, X. Yuan, and L. Gu: *Eng. Appl. Artif. Intell.* **124** (2023) 106563. <https://doi.org/10.1016/J.ENGAPPAI.2023.106563>
- 27 J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu: *IEEE Trans. Pattern Anal. Mach. Intell.* **42** (2017) 2011. <https://doi.org/10.1109/TPAMI.2019.2913372>
- 28 Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu: *Proc. 2019 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 11531. <https://doi.org/10.1109/CVPR42600.2020.01155>
- 29 S. Woo, J.Park, J. Y.Lee, and I. S. Kweon: *Lect. Notes Comput. Sci.* **11211** (2018) 3. https://doi.org/10.1007/978-3-030-01234-2_1
- 30 K. Simonyan and A. Zisserman: *Proc. 3rd Int. Conf. Learn. Represent (ICLR 2015)* <https://arxiv.org/abs/1409.1556v6> (accessed 26 March 2023).
- 31 K. He, X. Zhang, S. Ren, and J. Sun: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (IEEE, 2016)* 770–778. <https://doi.org/10.1109/CVPR.2016.90>

- 32 M. Tan and Q. V. Le: EfficientNet: 36th Int. Conf. Mach. Learn. (ICML, 2019) 2019 10691–10700 <https://arxiv.org/abs/1905.11946v5> (accessed 5 July 2023).
- 33 F. Chollet: Xception: Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognition (CVPR, 2017) 1800–1807. <https://doi.org/10.1109/CVPR.2017.195>
- 34 M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen: Proc. 2018 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (IEEE/CVS, 2018) 4510–4520. <https://doi.org/10.1109/CVPR.2018.00474>
- 35 C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna: Proc. 2016 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (IEEE, 2016) 2818–2826. <https://doi.org/10.1109/CVPR.2016.308>
- 36 J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl: Proc. 2011 Adv. Neural Inf. Process Syst. (NIPS, 2011) 24. <https://www.researchgate.net/publication/216816964>
- 37 G. Pattnaik, V. K. Shrivastava, and K. Parvathi: Appl. Artif. Intell. **34** (2020) 981. <https://doi.org/10.1080/08839514.2020.1792034>
- 38 L. Sun, K. Liang, Y. Wang, W. Zeng, X. Niu, and L. Jin: Soft Comput. **28** (2023) 1. <https://doi.org/10.1007/S00500-023-08631-W/METRICS>
- 39 Y. Chen, M. Chen, M. Guo, J. Wang, and N. Zheng: Front. Plant Sci. **14** (2023) 1282212. <https://doi.org/10.3389/FPLS.2023.1282212>
- 40 S. Jia, H. Gao, and X. Hang: J. Phys. Conf. Ser. **1437** (2020) 012052. <https://doi.org/10.1088/1742-6596/1437/1/012052>
- 41 H. Amin, A. Darwish, A. E. Hassanien, and M. Soliman: IEEE Access **10** (2022) 31103. <https://doi.org/10.1109/ACCESS.2022.3159678>
- 42 M. Agarwal, V. K. Bohat, M. D. Ansari, A. Sinha, S. K. Gupta, and D. Garg: Proc. 2019 IEEE 9th Int. Conf. Adv. Comput. (IACC, 2019) 176–181. <https://doi.org/10.1109/IACC48062.2019.8971602>
- 43 A. Waheed, M. Goyal, D. Gupta, A. Khanna, A. E. Hassanien, and H. M. Pandey: Comput. Electron. Agric. **175** (2020) 105456. <https://doi.org/10.1016/J.COMPAG.2020.105456>
- 44 S. Jasrotia, J. Yadav, N. Rajpal, M. Arora, and J. Chaudhary: Procedia Comput. Sci. **218** (2023) 1712. <https://doi.org/10.1016/J.PROCS.2023.01.149>
- 45 M. Li, G. Zhou, A. Chen, L. Li, and Y. Hu: Eng. Appl. Artif. Intell. **123** (2023) 106195. <https://doi.org/10.1016/J.ENGAPPAL.2023.106195>

About the Authors



Mei-Ling Huang received her M.S. and Ph.D. degrees in industrial engineering from the University of Wisconsin–Madison and National Chiao Tung University, respectively. Currently, she is affiliated with the Department of Industrial Engineering and Management at National Chin-Yi University of Technology. Her research interests include quality management, quality engineering, data mining, and medical diagnosis. (huangml@ncut.edu.tw)



You-An Chen received his B.S. degree in industrial engineering and management from National Chin-Yi University of Technology, Taichung, Taiwan, in 2022. Since 2022, he has been a graduate student in the Department of Industrial Engineering and Management, National Chin-Yi University of Technology. His research interests include quality engineering and data mining.