*sustainability*

MDPI

*Article*

# Plant Disease Classification and Adversarial Attack Using SimAM-EfficientNet and GP-MI-FGSM

**Haotian You, Yufang Lu * and Haihua Tang**

School of Information Science and Engineering, Guilin University of Technology, Guilin 541004, China
* Correspondence: luyufang165@163.com

**Abstract:** Plant diseases have received common attention, and deep learning has also been applied to plant diseases. Deep neural networks (DNNs) have achieved outstanding results in plant diseases. Furthermore, DNNs are very fragile, and adversarial attacks in image classification deserve much attention. It is important to detect the robustness of DNNs through adversarial attacks. The paper firstly improves the EfficientNet by adding the SimAM attention module. The SimAM-EfficientNet is proposed in this paper. The experimental results show that the accuracy of the improved model on PlantVillage reaches 99.31%. The accuracy of ResNet50 is 98.33%. The accuracy of ResNet18 is 98.31%. The accuracy of DenseNet is 98.90%. In addition, the GP-MI-FGSM adversarial attack algorithm improved by gamma correction and image pyramid in this paper can increase the success rate of attack. The model proposed in this paper has an error rate of 87.6% whenattacked by the GP-MI-FGSM adversarial attack algorithm. The success rate of GP-MI-FGSM proposed in this paper is higher than other adversarial attack algorithms, including FGSM, I-FGSM, and MI-FGSM.

**Keywords:** plant diseases; DNN; SimAM; efficientNet; FGSM; gamma correction; image pyramid



**Citation:** You, H.; Lu, Y.; Tang, H. Plant Disease Classification and Adversarial Attack Using SimAM-EfficientNet and GP-MI-FGSM. *Sustainability* **2023**, *15*, 1233. https://doi.org/10.3390/su15021233

Academic Editors: Hossein Bonakdari and Majid Sartaj

Received: 14 November 2022
Revised: 29 December 2022
Accepted: 3 January 2023
Published: 9 January 2023



## 1. Introduction

Agriculture produces food and other products through artificial cultivation. Plant diseases have received common attention, and deep learning has also been applied to plant diseases. Many kinds of plant diseases reduce the yield and quality of the crop. They bring losses to agricultural workers and have a bad influence on farmers. In addition, the efficiency of manual diagnosis of plant diseases is low. The accuracy rate may not be guaranteed. Therefore, the classification of plant diseases has received more attention from academic circles in recent years; it is a hot topic. Anshul used a genetic algorithm to classify plant diseases. Deep learning techniques are used to analyze diseases of tea, apple, tomato, grapevine, peach, and pear. Much progress has also been made in applying DNNs to classify plant diseases [1]. The exploration of artificial intelligence recognition technology started from the research of biological vision in the 1950s. Generally, the image capture equipment is used to automatically receive the target image and process. It also analyzes the image. It has the characteristics of fast speed, good stability, and high accuracy. In addition, the image capture equipment also has the potential to replace the human eye for recognition. In the 21st century, traditional machine learning methods and deep learning have been widely used in the research of artificial intelligence to identify agricultural pests and diseases. Early studies were based on static specimen images, and the recognition effect in the complex field environment needs to be improved. However, deep learning has certain advantages in processing massive data, which can automatically extract object features from large-scale data and use classifiers for classification and recognition. Compared with traditional machine learning, deep learning has a significant improvement in recognition accuracy and efficiency. It has significant advantages in improving recognition accuracy and reducing labor input. Therefore, the application of artificial intelligence in recognizing tea plant diseases and pests has great potential and demand. With the development

of artificial intelligence recognition technology in the identification system of agricultural pests and diseases, some progress has been made in the classification of plant diseases. The common neural networks are ResNet, VGG, and DenseNet [2–4]. Drumus used tomato images of the PlantVillage dataset to train many deep neural networks, and the accuracy of networks such as SqueezeNet improved a lot [5,6]. Shanwen Zhang identified the leaves of crape myrtle and other diseases and insect pests, and the classification accuracy reached 92.41% [7]. Lily proposed a simple and robust methodology for plant disease diagnosis using images in the visible spectrum of plants [8]. Kaur proposed the model DAG-ResNet in his paper. He used ECOC to identify many tomato diseases [9]. The accuracy was 98.8%. Wenliang Tang used conditional convolution, channel attention module, and knowledge distillation to improve the model, and the average accuracy reached 97.7% [10]. In this paper, the no-parameter attention SimAM is integrated into EfficientNet [11,12], and the improved SimAM-EfficientNet has the advantages of smaller parameters and shorter training time. The accuracy reaches 99.6%. The security issues of DNN have begun to receive attention because of the vulnerability of deep neural networks. Current studies have shown that DNNs are very susceptible to the interference of adversarial examples, and they make misjudgments. Adversarial examples are formed by adding carefully designed small perturbations to normal examples. These adversarial examples are used to fool humans and various DNNs. According to the circumstances, whether the internal structure of the target model is transparent, adversarial attack algorithms have two types: black-box attack and white-box attack. The white-box attack algorithms include FGSM, DeepFool, JSMA, C&W, etc. [13–16]. The black-box attack algorithms currently include single-pixel attack algorithms, local search attack algorithms, and other algorithms [17]. FGSM and other adversarial attack algorithms are based on gradients. They use gradients to add interference to normal examples and effectively generate adversarial examples to interfere with various deep neural networks. GP-MI-FGSM incorporates image pyramid and histogram normalization in the field of image enhancement into adversarial attack algorithms based on gradients. The adversarial attack algorithm can generate adversarial examples with higher attack success rates, and it is universal to various deep neural networks. In addition, the perturbation is so small that people can hardly detect it. At present, the classification of plant disease can still be improved, so we propose a new DNN called SimAM-EfficientNet. It basically adds the SimAM attention module into EfficientNet. A number of experiments prove that our network model has a good performance. In addition, our team also designed a new adversarial example algorithm called GP-MI-FGSM. The adversarial examples generated by this algorithm can also be used to train DNNs. Training with adversarial examples will make DNNs more robust.

## 2. Deep Neural Network

### *2.1. Convolutional Neural Network*

Deep learning provides various models and algorithms to process data as efficiently as the human brain's biological nervous systems or neuronal responses [18]. A convolutional neural network contains a convolutional structure. It includes five parts: input layer, convolutional layer, pooling layer, fully connected layer, and output layer. Compared with traditional neural networks, a convolutional structure can reduce the amount of memory occupied by the deep network, effectively reducing the network's complexity [19]. Many convolutional neural networks, such as VGG, ResNet, and DenseNet, are commonly used to classify pests and diseases [20–22]. In this paper, the improved EfficientNet is used for classification, and it is fast and accurate.

### *2.2. EfficientNet*

In May 2019, Google proposed a new convolutional neural network named EfficientNet. Although the current network has made good progress in accuracy and speed, the author uses a multidimensional hybrid model scaling method to make the model consider both speed and accuracy. ResNet mainly increases the depth of the network to im-

prove accuracy. EfficientNet balances the network depth, width, and resolution through compound scaling factors, ensuring speed and improving accuracy. The first EfficientNet model, EfficientNet-B0, is structured as shown in Table 1. B0 is the most basic model. B1 to B7 are seven models modified from B0 in terms of channels, layers, and resolution.

**Table 1.** EfficientNet-B0 structure.

| Stage | Operator | Channels | Layers | Resolution |
|---|---|---|---|---|
| 1 | Conv3 × 3 | 32 | 1 | 224 × 224 |
| 2 | MBConv1,k3 × 3 | 16 | 1 | 112 × 112 |
| 3 | MBConv6,k3 × 3 | 24 | 2 | 112 × 112 |
| 4 | MBConv6,k5 × 5 | 40 | 2 | 56 × 56 |
| 5 | MBConv6,k3 × 3 | 80 | 3 | 28 × 28 |
| 6 | MBConv6,k5 × 5 | 112 | 3 | 14 × 14 |
| 7 | MBConv6,k5 × 5 | 192 | 4 | 14 × 14 |
| 8 | MBConv6,k3 × 3 | 320 | 1 | 7 × 7 |
| 9 | Conv1 × 1 & Pooling & FC | 1280 | 1 | 7 × 7 |

### *2.3. SimAM Attention Module*

A lot of current attention modules generate 1D or 2D weights. Then, the generated weights are expanded for channel and spatial attention. Most current attention modules generally have the following two problems. The first is that these attention modules can only extract features by channel and space, which leads to the inflexibility of attention weights. In addition, the structures of various convolutional neural networks are complex and are influenced by a series of factors. In contrast to them, SimAM considers both space and channel. It proposes three-dimensional attention weights without adding parameters to the original network. In detail, it defines an energy function based on neuroscience theory and derives a solution that can be quickly converged. This process can be implemented in 10 lines of code. Another advantage of SimAM is that it avoids too much adjustment to the network structure. Therefore, SimAM is more flexible, modular, and lightweight. In many cases, SimAM is superior to the most representative SE and CBAM attention modules. As shown in Figure 1, SimAM can outperform these two popular kinds of attention modules [23,24].

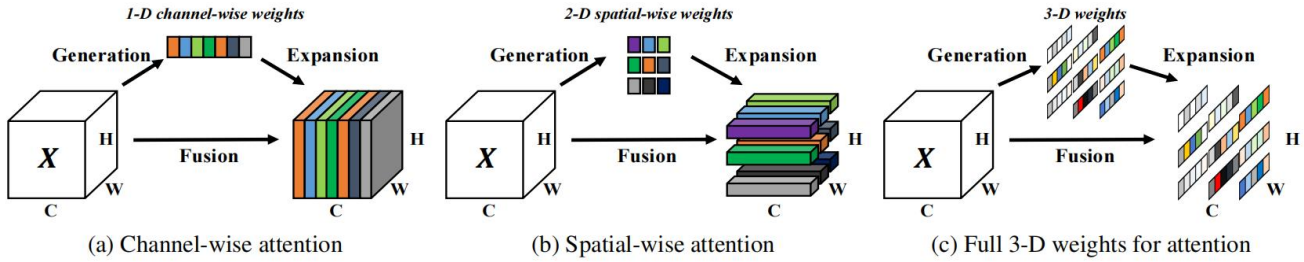


**Figure 1.** Comparisons of three kinds of attention steps.

The SimAM module looks for important neurons and defines an energy function. It uses binary labels and adds regular terms. Finally, the minimal energy can be calculated with the following:

$$e_t^* = \left(4\left(\lambda+\sigma^2\right)\right) / \left((t-u)^2 + 2\sigma^2 + 2\lambda\right) \quad (1)$$

where

$$u_t = \frac{1}{M-1}\sum_{i=1}^{M-1} x_i, \sigma_t^2 = \frac{1}{M-1}\sum_{i=1}^{M-1}\left(x_i - u_t\right)^2 \quad (2)$$

Among them, $\mu_t$ is the mean of all neurons. $\sigma_t^2$ is the variance of all neurons. $t$ is the target neuron. $x_i$ are other neurons in a channel of the input feature. $\lambda$ is the regularization coefficient. The number of neurons on that channel can be obtained by $M = H \times W$. In conclusion, the difference between neurons and peripheral neurons is related to energy. The significance of each neuron can be calculated by $1/e^*$. It uses a scaling operator to refine features. The entire refinement phase of the SimAM module is:

$$X = X \bullet sigmoid(1/E) \tag{3}$$

where $E$ groups all $e$ across the channel and spatial dimensions. We use *sigmoid* to limit the size of value in $E$.

### 2.4. SimAM-EfficientNet

The basic structure of EfficientNet is shown in Table 1, and EfficientNet-B0 has a total of nine stages. The first stage is a 3 × 3 convolution layer. The second to eighth stages are MBConv, which is also the most important structure of this network model. The last stage is composed of a 1 × 1 convolution layer, a pooling layer, and the last fully connected layer. MBConv consists of five parts. The first part is a 1 × 1 convolution layer. The second part is a depthwise convolution, followed by the SE attention module. The fourth part is a 1 × 1 convolution layer for dimensionality reduction. Finally, the last is a dropout layer to alleviate overfitting problems. The SimAM module was added after the first convolutional layer to increase channel and spatial weights. The basic structure in this paper is shown in Figure 2, namely, MBConv. The rest of the structure is appropriately modified by it. The original EfficientNet includes the SE attention module. Our model adds the SimAM attention module to improve it.

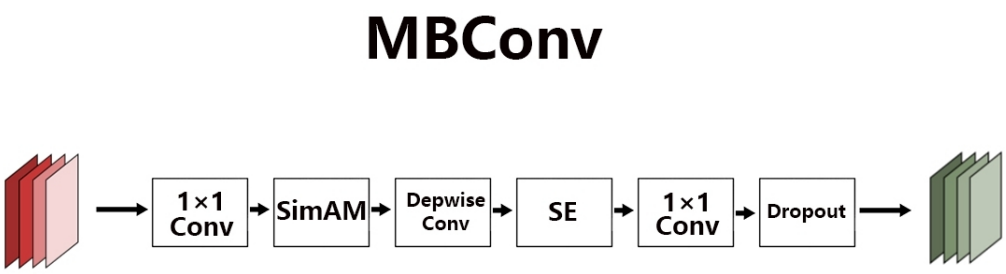


**Figure 2.** Modified MBConv.

As shown in Figure 3, the final SimAM-EfficientNet can be obtained after modifying MBConv.

In this paper, SimAM-EfficientNet is composed of seven improved SimAM-MBConv modules, two convolutional layers, one pooling layer, and one fully connected layer. In Figure 3, different colors and sizes represent different layers. To begin with, the images with dimensions of 224 × 224 × 3 are ascended by the 3 × 3 convolutional layer. The dimensions of the obtained images with features are 112 × 112 × 32. The next step is extracting the images' features using SimAM-Conv. If two SimAM-Convs are the same, the connection will be deactivated, and the input will connect. After 1 × 1 point by point convolution, the original channel is restored, and the fully connected layer is used for classification.

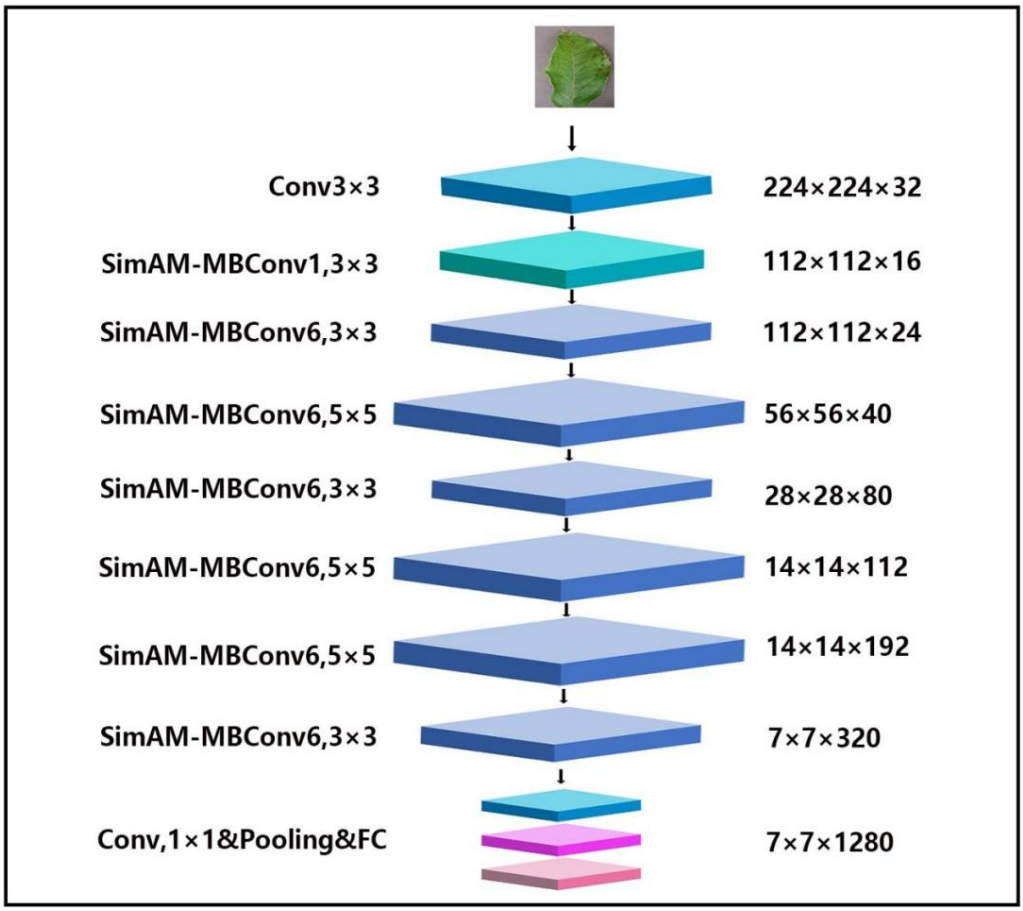


**Figure 3.** SimAM-EfficientNet structure.

## 3. Adversarial Examples

### *3.1. Adversarial Attack*

Adversarial examples are designed to attack DNNs. They are almost indistinguishable from normal examples but can lead to model errors. The existence of adversarial examples threatens the deep neural networks and affects the security of related fields. The existing adversarial attack algorithms include white-box attack algorithms and black-box attack algorithms [25,26]. White-box attack algorithms require detailed information about the attacked models, such as gradients. At present, the white-box attack algorithms have also achieved good results. Black-box attack algorithms do not know the information of the attacked model.

### *3.2. FGSM*

FGSM (Fast Gradient Sign Method) is an algorithm based on gradient generation of adversarial examples. It was presented by Goodfellow at the ICLR2015 conference. After that, improved I-FGSM and MI-FGSM appeared successively. The main reason for the low attack success rate of current methods is the phenomenon of overfitting in the generation of adversarial examples. Adversarial examples for one model are less successful when used for other models. Therefore, the image enhancement technology of the training model is integrated into adversarial examples in this paper. The overfitting phenomenon is alleviated, and the attack success rate and transferability are improved.

### *3.3. GP-MI-FGSM*

#### 3.3.1. Formulas

In this paper, a new adversarial attack algorithm based on image pyramid and histogram normalization is proposed by using image enhancement technology. Image pyra-

mids are divided into the Laplacian pyramid and the Gaussian pyramid. The algorithm can resize the images. This algorithm improves the diversity of input images through image pyramid and histogram normalization. It effectively alleviates the overfitting phenomenon of adversarial attacks and improves the attack success rate of adversarial examples. FGSM is the most basic adversarial attack algorithm. Adversarial examples can be obtained by:

$$x^{adv} = x + \varepsilon \bullet sign(\nabla_x J(x, y)) \tag{4}$$

In the formula, $sign()$ is a symbolic function and $\nabla_x J(x, y)$ represents a gradient. $\varepsilon$ makes the disturbance satisfy the norm constraint. I-FGSM is an iterative version of FGSM. It decomposes a single step into multiple steps to achieve less disturbance. Adversarial examples can be obtained by:

$$x_0^{adv} = x, x_{t+1}^{adv} = x_t^{adv} + \alpha \bullet sign(\nabla_{x_t^{adv}} J(x_t^{adv}, y)) \tag{5}$$

In the formula, $x_{t+1}^{adv}$ is the adversarial example generated by iterating $t+1$ times. $T$ represents the number of iterations. The step length can be obtained by $\alpha = \varepsilon/T$. It ensures that the adversarial examples generated are in the neighborhood of $x$. $\varepsilon$ is the size of the neighborhood. MI-FGSM is based on I-FGSM and adds a momentum factor. In each iteration, the gradient information of each previous iteration is added to stabilize the update direction and avoid falling into the local extremum. The update process can be described by the following:

$$g_{t+1} = \mu \bullet g_t + (\nabla_{x_t^{adv}} J(x_t^{adv}, y))/\|(\nabla_{x_t^{adv}} J(x_t^{adv}, y))\|_1 \tag{6}$$

$$x_{t+1}^{adv} = Clip_x^{\varepsilon} x_t^{adv} + \alpha \bullet sign(g_{t+1}) \tag{7}$$

In the formula, $g_{t+1}$ means that the cumulative gradient iterated $t+1$ times, and $\mu$ is the momentum decay factor. Therefore, this paper effectively integrates image pyramid and histogram normalization with MI-FGSM to become a better adversarial attack algorithm. $D$ is the image enhancement function. It includes image pyramid and histogram normalization. The final function is given by:

$$g_{t+1} = \mu \bullet g_t + (\nabla_{x_t^{adv}} J(D(x_t^{adv}), y))/\|(\nabla_{x_t^{adv}} J(D(x_t^{adv}), y))\|_1 \tag{8}$$

#### 3.3.2. Concrete Steps

As shown in Figure 4, the original adversarial examples generate the adversarial examples through the adversarial attack algorithm. Firstly, the example is processed by histogram normalization and image pyramid, and a 224 × 224 × 3 image is obtained. Then, the obtained image is input into the model, and the loss function is calculated. The example is updated iteratively along the gradient of the loss function, and then the perturbation is added. If the requirements are not met, iterations continue until it succeeds. Finally, the adversarial example is output. The final GP-MI-FGSM algorithm is divided into several steps.

To begin with, several inputs are required. $x$ is a original example and its actual label is denoted as $y^{true}$. Deep neural networks are expressed as $f$. $L$ is a loss function. $D$ is a image enhancement function. $\varepsilon$ represents the maximum disturbance value. $T$ is the maximum number of iterations. The decay factor is denoted as $\mu$. Then the perturbation is calculated by $\alpha = \varepsilon/T$. $x_0^{adv}$ is initialized to $x$. $g_0$ is initialized to 0. After iterations, the example $x_t^{adv}$ can be obtained by $D(x_t^{adv})$. The next step is to calculate the loss function $\nabla_{x_t^{adv}} J(D(x_t^{adv}), y)$. Futhermore, $g_{t+1}$ and $x_{t+1}^{adv}$ are obtained. Finally, the value of $X_T^{adv}$ is returned.

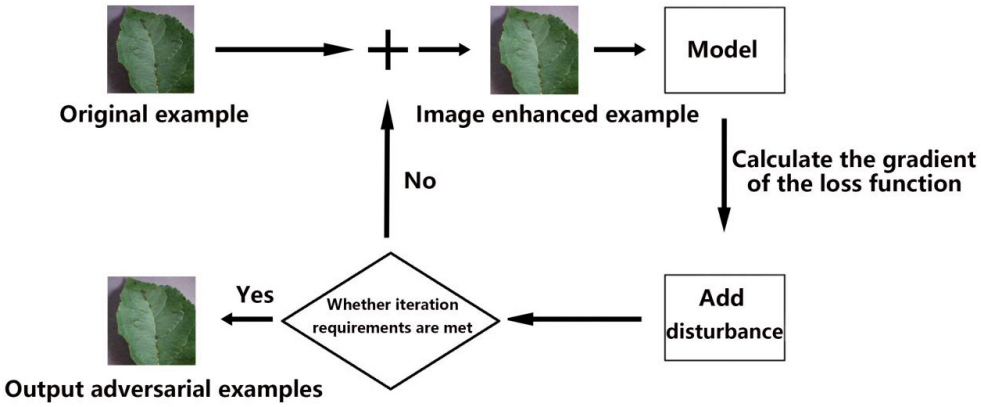


**Figure 4.** Using GP-MI-FGSM attack models.

## 4. Model Training Strategy

### *4.1. Learning Rate Decay*

In the early stage of model training, the larger the learning rate, the longer the step length. Then the gradient can move down at a faster rate. However, in the later stage of model training, the strategy of gradually reducing the learning rate and the step length is adopted. It will help the algorithm converge and easily find the optimal solution. Thus, the experiments in this paper add learning rate decay to the training process. According to the training situation without adding learning rate decay, the method of exponential decay is finally chosen. As shown in Figure 5, the learning rate will decline exponentially.

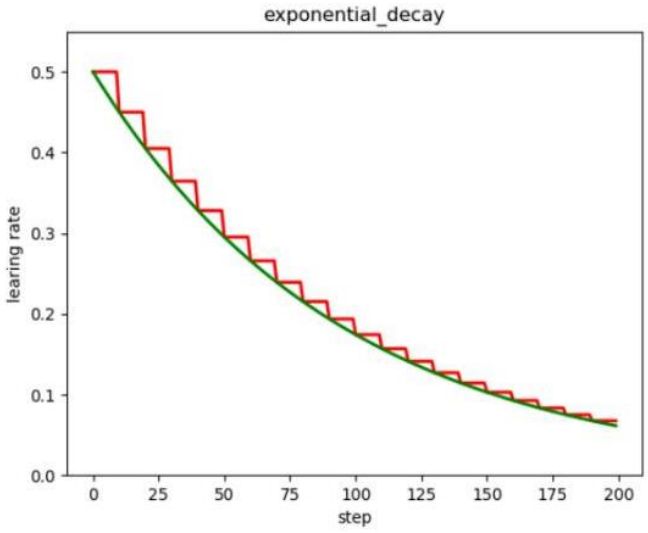


**Figure 5.** Exponential decay.

### *4.2. Momentum*

In classical mechanics, momentum is the product of mass and velocity [27]. It is a physical quantity related to the mass and velocity of an object. The stochastic gradient descent optimizer (SGD) has a problem with oscillations caused by updates that do not fully use curvature information [28]. It causes SGD to slow down when the curvature is high. The faster optimization path can be obtained by using the average gradient. It helps suppress oscillations because the gradient in the opposite direction is canceled out. Momentum causes a change in the opposite direction, which leads to vibration attenuation on surfaces with high curvature. If the momentum term is big, the learning rate should remain small. Large values of momentum also mean that convergence will occur quickly; however, if both momentum and learning rate remain big, minimum should be skipped by leaps and bounds. Small momentum values cannot effectively avoid local minima.

It also makes training slow. If the gradient keeps changing direction, momentum will also help smooth the change.

### *4.3. Weight Decay*

Although enlarging datasets may alleviate overfitting, acquiring additional data is often difficult. Weight decay is a common method to deal with overfitting problems. In a sense, smaller weights indicate lower complexity of the network and better data fitting, which has also been verified in practical applications. The regularization effect of L2 is often better than that of nonregularization. When overfitting, the coefficient of the fitting function is generally large. Overfitting means that the fitting function needs to consider every point, and the final fitting function fluctuates greatly. The value of a function can change dramatically in a very short period of time. It signifies that the derivative value of the function in some cells is very large. Because the value of the independent variable is uncertain, only the coefficient is large enough to ensure a big derivative value. Regularization restricts the norm of parameters, which are not too large. The overfitting situation can be reduced to a certain extent.

### *4.4. Transfer Learning*

Many deep neural networks trained on natural images show a strange phenomenon at the first layer, where they learn features similar to filters and color blobs [29]. These first-layer features do not appear specific to a particular dataset or task. Instead, they apply to many datasets and tasks. Finally, features must transition from general to specific by the last layer of the deep neural network. Transfer learning stores the knowledge or model learned in one domain and applies it to other related domains or problems. For example, the knowledge used to identify cars can also be used to improve the ability to identify trucks. The underlying characteristics of image datasets in different domains are similar. The transfer learning method improves learning efficiency through the common features of the convolution layer and knowledge transfer. The learning process is relatively stable. Training with a large ImageNet dataset, EfficientNet will obtain initial weights. Then, the trained weights are transferred to the SimAM-EfficientNet model for parameter initialization. Finally, the PlantVillage dataset is used for optimization learning, and fine-tuning model parameters enhance the learning ability of SimAM-EfficientNet. Compared with random weight initialization, weight initialization using ImageNet to pretrain can greatly accelerate the model's convergence and improve the model's generalization ability.

## 5. Experimental Results and Analysis

### *5.1. Classification*

#### 5.1.1. Experimental Environment and Parameter Settings

All experiments used the same environment with 32 GB RAM, GPU NVDIA GeForce 2070, deep learning framework Pytorch, and 500 GB hard drive. In terms of parameter settings of model training, the experiments adopted the batch training method that divides the training set and test set into multiple batches. The batch size was set to 16. The number of iterations was 50 rounds. The stochastic gradient descent (SGD) algorithm optimized the model during the training process. The experimental learning rate was set to 0.001, and the dropout was 0.0005. The exponential decay of learning rate was adopted in the experiment, and gamma was set to 0.9.

#### 5.1.2. Experimental Data Source

The experiment used the PlantVillage dataset to train models. The dataset contains 54,303 health and disease images, divided into 38 categories. The dataset was divided into training, validation, and test sets by 90%, 7%, and 3%.

5.1.3. Performance Metrics

Multiple classifications are required because the PlantVillage dataset has 39 categories. The definitions of indices are given between Equations (9)–(16). They include true positive (*TP*), true negative (*TN*), false positive (*FP*), and false negative (*FN*). *TP* is the number of correctly classified unhealthy images. *TN* is the number of the correctly classified images in all other categories except for the correlative class. *FN* is the number of incorrectly classified images from the correlative class. *FP* is the number of incorrectly classified images in all other categories except for the correlative class.

$$Sen = TP/(TP + FN) \tag{9}$$

$$Spe = TN/(TN + FP) \tag{10}$$

$$Acc = (TP + TN)/(TP + FN + TN + FP) \tag{11}$$

$$Pre = TP/(TP + FP) \tag{12}$$

$$AveragenSen = \frac{1}{classes}\sum_{1}^{classes} Sen \tag{13}$$

$$AveragenSpe = \frac{1}{classes}\sum_{1}^{classes} Spe \tag{14}$$

$$AveragenAcc = \frac{1}{classes}\sum_{1}^{classes} Acc \tag{15}$$

$$AveragenPre = \frac{1}{classes}\sum_{1}^{classes} Pre \tag{16}$$

5.1.4. Results

In order to verify the performance advantages of the proposed SimAM-EfficientNet model, the PlantVillage dataset was used to train multiple datasets in a single experimental environment. The models include ResNet18, ResNet50, DenseNet121, VGG16, EfficientNet, and SimAM-EfficientNet (our model). As seen in Table 2, the accuracy of SimAM-EfficientNetB0 was 99.31%, 2.5% higher than the original model. The best remaining model, DenseNet121, achieved 98.90% accuracy, only 4.1% lower than our model. Two members of ResNet performed poorly in the experiment. The accuracy of ResNet18 was 98.31%, 0.02% lower than ResNet50. VGG16 had an average performance and 98.51% accuracy. Finally, a pretraining strategy was added to improve the recognition accuracy. It improved the accuracy to 99.47%. Efficiency is also important when deep learning networks perform recognition tasks. The number of parameters is an important index to measure efficiency. The table shows that the number of parameters varies greatly between models. VGG16 has the largest number of parameters and runs slowly. Resnet50 has 23.5 M parameters, about 100 M less than VGG16. ResNet18 has a more streamlined structure, with half the number of parameters of ResNet50. DenseNet121 is an improvement of ResNet and has few parameters compared to the previously mentioned models. Finally, SimAM-EfficientNetB0 has the lowest number of parameters, about 30 times fewer than VGG16. Its number of parameters is 3.83 M. The experimental results showed that the time per epoch of SimAM-EfficientNetB0 was still the shortest. Under the same environment and configuration, the time per epoch of SimAM-EfficientNetB0 during the training takes only 642 s, which is about half that of ResNet18. VGG16 has a large number of parameters and takes up to 2880 s per epoch. The time per epoch of DenseNet121 is 1560 s. In addition, SimAM-EfficientNetB0 takes

an average of 1.12 s to detect a single image. It takes less time than other models. Compared with the original model, the proposed model in this paper does not add any parameters. The model in this paper considers both efficiency and accuracy.

**Table 2.** Comparative experiments of different models.

| Model | Avg Acc (%) | Avg Sen (%) | Avg Spe (%) | Avg Pre (%) | Time per Epoch (s) | Parameters (M) | Time Cost (s) |
|---|---|---|---|---|---|---|---|
| SimAM-EfficientNetB0 | 99.31% | 97.92% | 99.61% | 98.29% | 642 | 3.83 M | 1.12 |
| SimAM-EfficientNetB1 | 98.32% | 96.11% | 99.35% | 96.11% | 980 | 6.22 M | 1.18 |
| SimAM-EfficientNetB2 | 98.29% | 96.72% | 99.41% | 96.25% | 991 | 8.31 M | 1.21 |
| SimAM-EfficientNetB3 | 98.29% | 96.77% | 99.81% | 96.39% | 1190 | 10.22 M | 1.25 |
| SimAM-EfficientNetB4 | 99.22% | 96.82% | 99.33% | 97.31% | 1680 | 16.75 M | 1.28 |
| SimAM-EfficientNetB5 | 99.15% | 97.11% | 99.28% | 96.77% | 2270 | 27.05 M | 1.42 |
| SimAM-EfficientNetB6 | 99.27% | 95.16% | 99.39% | 96.38% | 2880 | 38.87 M | 1.58 |
| SimAM-EfficientNetB7 | 99.31% | 96.88% | 99.20% | 95.22% | 3580 | 60.86 M | 1.66 |
| SimAM-EfficientNetB0 (pretrained) | 99.47% | 98.11% | 99.89% | 98.78% | 642 | 3.83 M | 1.12 |
| ResNet18 | 98.31% | 93.54% | 99.32% | 96.45% | 1150 | 11.2 M | 1.14 |
| ResNet50 | 98.33% | 95.49% | 99.84% | 95.31% | 1780 | 23.5 M | 1.28 |
| VGG16 | 98.51% | 94.21% | 99.25% | 96.77% | 2880 | 134.4 M | 1.68 |
| DenseNet121 | 98.90% | 97.87% | 99.71% | 97.92% | 1560 | 6.9 M | 1.25 |
| EfficientNetB0 | 99.06% | 96.81% | 99.77% | 97.29% | 642 | 3.83 M | 1.12 |

As seen in Table 3, the eight indices measure the performance of SimAM-EfficientNet on 38 different classes. The model proposed in this paper classifies images successfully in most cases, but there are some errors in Corn Cercospo, Peach Bacterial spot, and Tomato Septoria leaf spot.

**Table 3.** Classification performance of SimAM-EfficientNetB0.

| Class | TP | TN | FP | FN | Acc (%) | Sen (%) | Spe (%) | Pre (%) |
|---|---|---|---|---|---|---|---|---|
| Apple scab | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Apple Black rot | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Apple Cedar apple rust | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Apple healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Background without leaves | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Blueberry healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Cherry Powdery mildew | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Cherry healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Corn Cercospora leaf spot Gray leaf spot | 50 | 1898 | 2 | 0 | 99.9 | 100 | 99.89 | 96.15 |
| Corn Common rust | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Corn Northern Leaf Blight | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Corn healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Grape Black rot | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Grape Esca (Black Measles) | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Grape Leaf blight | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Grape healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |

**Table 3.** *Cont.*

| Class | TP | TN | FP | FN | Acc (%) | Sen (%) | Spe (%) | Pre (%) |
|---|---|---|---|---|---|---|---|---|
| Orange Haunglongbing | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Peach Bacterial spot | 50 | 1898 | 2 | 0 | 99.9 | 100 | 99.89 | 96.15 |
| Peach healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Pepper, bell Bacterial spot | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Pepper, bell healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Potato Early blight | 49 | 1899 | 0 | 1 | 99.95 | 98 | 100 | 100 |
| Potato Late blight | 49 | 1899 | 0 | 1 | 99.95 | 98 | 100 | 100 |
| Potato healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Raspberry healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Soybean healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Squash Powdery mildew | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Strawberry Leaf scorch | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Strawberry healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Bacterial spot | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Early blight | 49 | 1899 | 0 | 1 | 99.95 | 98 | 100 | 100 |
| Tomato Late blight | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Leaf Mold | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Septoria leaf spot | 50 | 1898 | 2 | 0 | 99.9 | 100 | 99.89 | 96.15 |
| Tomato mites | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Target Spot | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato Leaf Curl Virus | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato mosaic virus | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |
| Tomato healthy | 50 | 1900 | 0 | 0 | 100 | 100 | 100 | 100 |

As shown in Table 2, the performance of SimAM-EfficientNet proposed in this paper is superior to other models. The specific training process is shown in Figures 6–11.

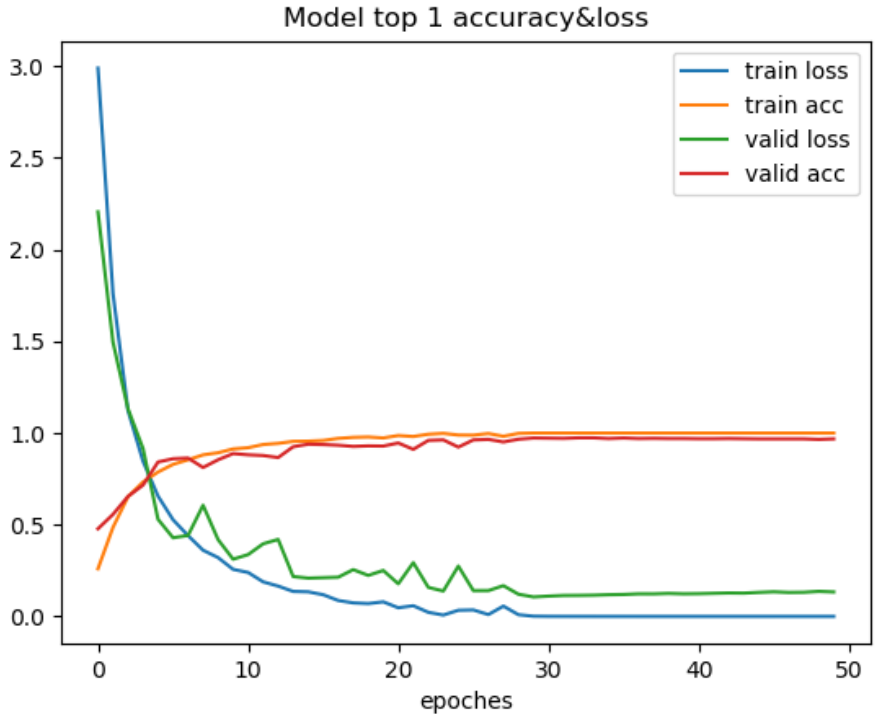


**Figure 6.** Changes in recognition accuracy and loss values during training of ResNet50.

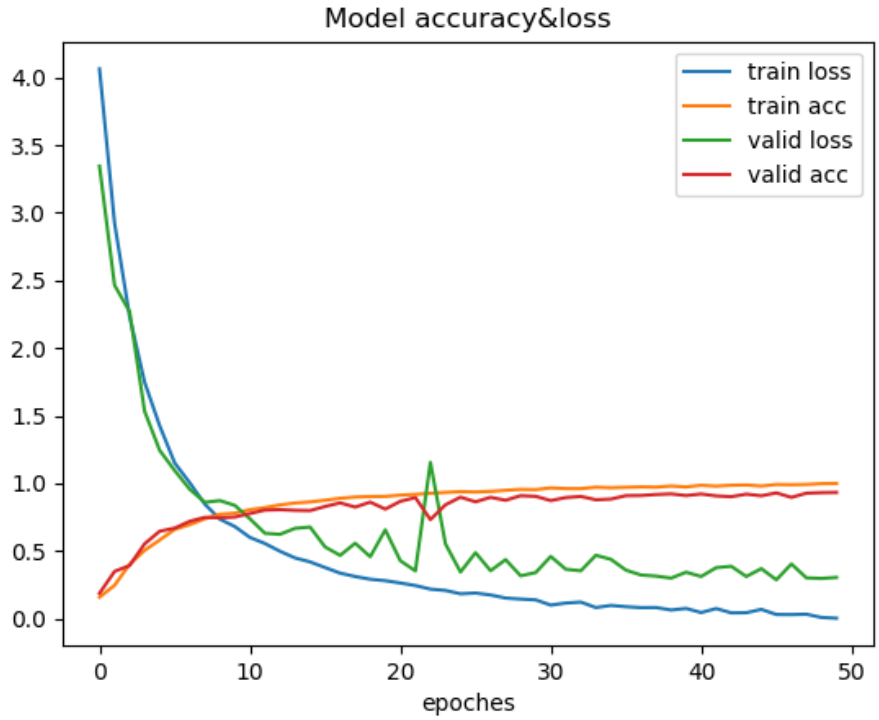


**Figure 7.** Changes in recognition accuracy and loss values during training of ResNet18.

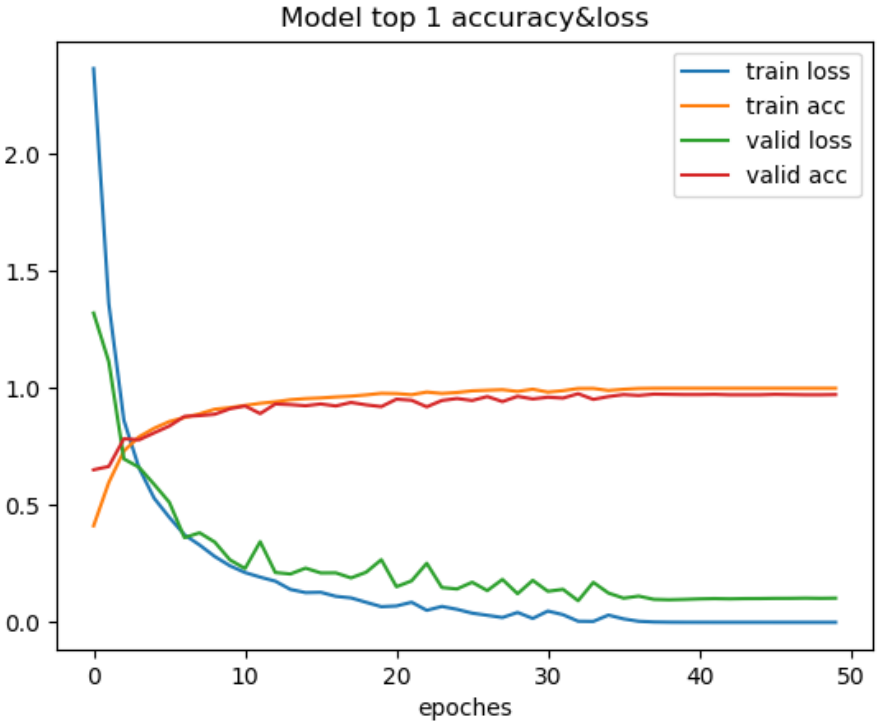


**Figure 8.** Changes in recognition accuracy and loss values during training of DenseNet121.

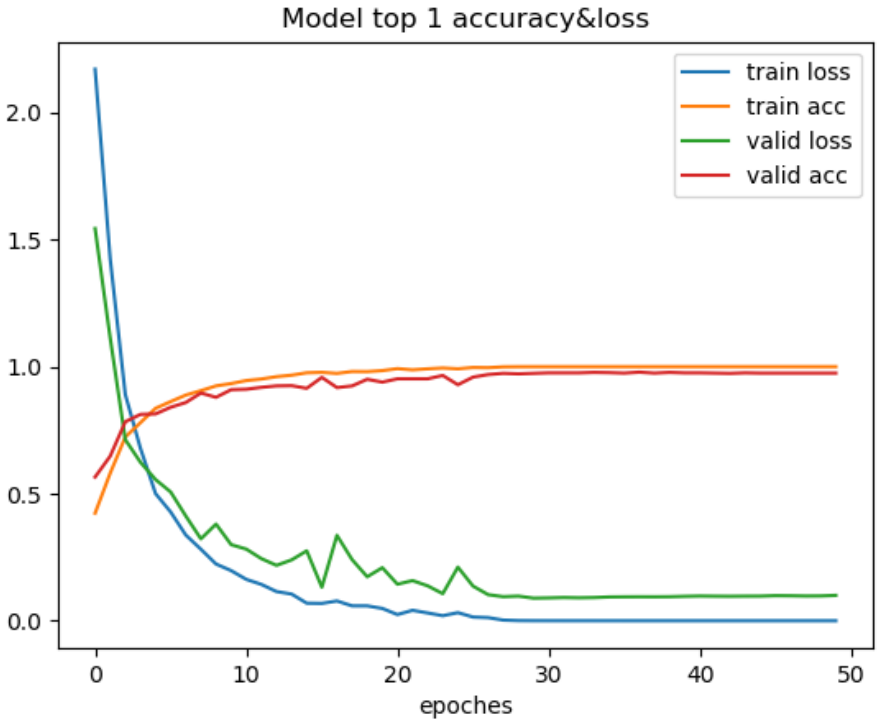


**Figure 9.** Changes in recognition accuracy and loss values during training of VGG16.

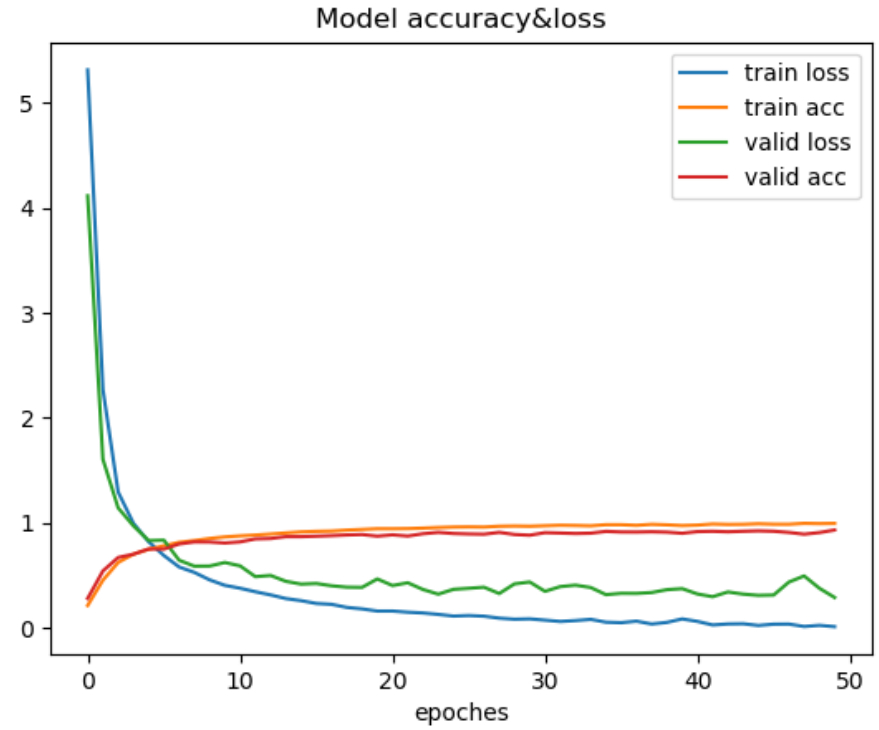


**Figure 10.** Changes in recognition accuracy and loss values during training of EfficientNet.

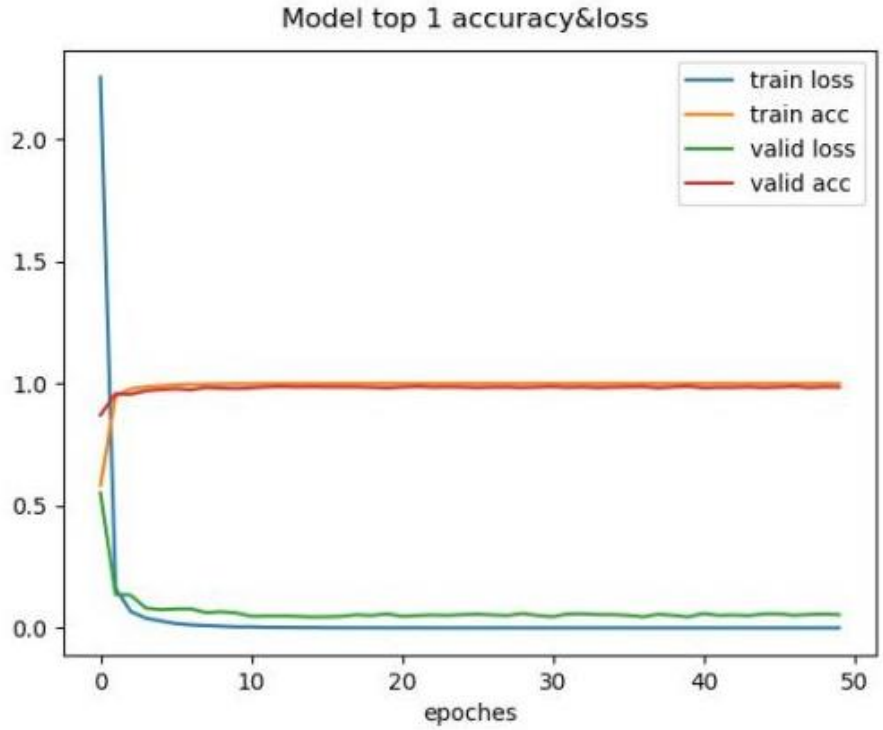


**Figure 11.** Changes in recognition accuracy and loss values during training of SimAM-EfficientNetB0.

Through the comparison of the above seven training processes, the results can be obtained. The convergence process of SimAM-EfficientNet is smooth while ensuring speed and accuracy. In this paper, the transfer learning strategy is added to improve convergence speed. To begin with, ImageNet is used to pretrain the model in this paper. Then, the pretrained model is trained with the PlantVillage dataset. The improved model in this paper has many advantages. Compared with other models, the model in this paper has the smallest training loss and the fastest convergence speed. After more than six iterations, the loss value tends to be stable. It is verified that the improved model can be better applied to the field of pest classification.

### *5.2. Experiment of Adversarial Example*

#### 5.2.1. Experimental Data Source

PlantVillage is still used for the dataset. The models include ResNet, DenseNet, VGG16, EfficientNet, and Simam-EfficientNet. The dataset contains 54,303 health and disease images, divided into 38 categories.

#### 5.2.2. The Benchmark Method

In order to comprehensively and fairly prove the effectiveness of the adversarial attack algorithm proposed in this paper, the experiment includes two other classical white-box antiattack methods (I-FGSM and MI-FGSM).

#### 5.2.3. Hyperparameter Settings

The study did not set the hyperparameters exactly as the norm in the momentum method, but, instead, chose smaller perturbations, in order to make the perturbations more difficult to detect. Maximum disturbance $\varepsilon = 0.3$; iteration times $T = 10$; step size $\alpha = 0.03$; attenuation factor $\mu = 1.0$.

#### 5.2.4. Experimental Results

The adversarial examples and original examples generated by the experiment are shown in Figure 12. Various algorithms were used to generate adversarial examples.

The adversarial examples generated by GP-MI-FGSM proposed in this paper differ little from the original examples and are hard to distinguish with human eyes. Compared with many existing adversarial attack algorithms, the maximum perturbation was only set to 0.3. The experiment used subtle perturbations, and the success rate of the attack algorithm increased. As shown in Table 3, the experiment compares four white-box adversarial algorithms, including FGSM, I-FGSM, MI-FGSM, and GP-MI-FGSM proposed in this paper. VGG16, ResNet50, DenseNet121, ResNet18, EfficientNet, and SimAm-EfficientNet were all attacked with these four adversarial attack algorithms. The experimental results show that all five models are vulnerable to adversarial attacks. The recognition accuracy of the proposed model also decreased by about 91%.

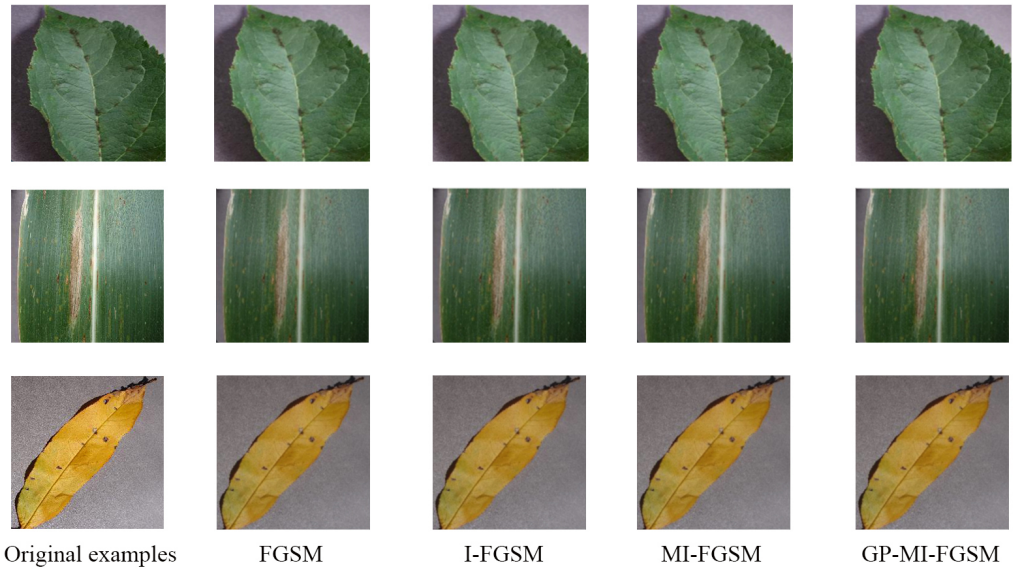


**Figure 12.** Adversarial examples generated by various algorithms.

As shown in Table 4, VGG16 performs the worst among these models. The success rate can reach 99.2% using the adversarial attack algorithm proposed in this paper. The next-worst performer is Resnet50, which has a recognition rate of only 2.9% when attacked by this algorithm. DenseNet was 1.8 percent more accurate than ResNet50. The algorithm's EfficientNet attack success rate was 92.1%, 1.5% higher than SimAM-EfficientNetB0. The table also shows the gap among the four adversarial algorithms. In the experiment of VGG16, the attack success rate of GP-MI-FGSM was 6.7% higher than FGSM. In the experiments using the model proposed in this paper, the difference was 4.1%. It can be seen that the GP-MI-FGSM adversarial algorithm proposed in this paper has the highest attack success rate.

**Table 4.** Comparison of attack success rates of different adversarial attack algorithms.

| Algorithms | VGG16 | ResNet50 | DenseNet | EfficientNet | SimAM-EfficientNetB0 |
|---|---|---|---|---|---|
| FGSM | 92.5% | 88.1% | 87.2% | 86.9% | 86.5% |
| I-FGSM | 92.6% | 89.3% | 88.5% | 87.8% | 86.8% |
| MI-FGSM | 93.1% | 90.2% | 89.9% | 89.3% | 88.7% |
| GP-MI-FGSM | 99.2% | 97.1% | 95.3% | 92.1% | 90.6% |

Adversarial examples can enhance the generalization ability and performance. Therefore, 10,000 adversarial examples generated by GP-MI-FGSM were added to the training set. After adversarial training, the accuracy of each model is shown in Table 5. The pre-trained SimAM-EfficientNet achieves an accuracy of 99.78%. The accuracy is 0.31% higher than the training without adversarial examples. The remaining models also show some

performance gains. The experimental results show that the adversarial examples generated by GP-MI-FGSM can improve the deep learning models.

**Table 5.** The experiment results on the Image VID dataset.

| Models | mAP (%) | mAP (%) (Slow) | mAP (%) (Medium) | mAP (%) (Fast) |
|---|---|---|---|---|
| SimAM-EfficientNetB0 (adversarial trained) | 99.59% | 98.22% | 99.83% | 98.59% |
| SimAM-EfficientNetB0 (adversarial trained and pretrained) | 99.78% | 99.05% | 99.98% | 99.52% |
| ResNet18 (adversarial trained) | 98.63% | 93.74% | 99.61% | 96.71% |
| ResNet50 (adversarial trained) | 98.61% | 95.72% | 99.87% | 95.73% |
| VGG16 (adversarial trained) | 98.72% | 94.63% | 99.66% | 96.93% |
| DenseNet121 (adversarial trained) | 99.28% | 98.15% | 99.82% | 98.24% |
| EfficientNetB0 (adversarial trained) | 99.32% | 97.21% | 99.89% | 97.79% |

## 6. Conclusions

This paper proposes the SimAM-EfficientNet based on the integration of 3D attention SimAM with MBConv. SimAM-EfficientNet includes eight models and the best one is SimAM-EfficientB0. Currently, SE, CBAM, and other attention can only refine features through channel and space, leading to inflexible attention weights. Because models are complex, more dimensions need to be considered. In contrast to these attention modules, SimAM considers both space and channel. The attention weights of three dimensions are deduced without adding parameters. In this paper, comparative experiments of multiple models were carried out to analyze the model recognition rate and required parameters. The model proposed in this paper was applied to classify pests and diseases. Histogram normalization and image pyramid were integrated into the MI-FGSM adversarial attack algorithm, and the new algorithm GP-MI-FGSM was obtained. Experiments included adversarial attacks on different models using various algorithms. The conclusion is as follows:

1. Compared with VGG16, ResNet50, DenseNet, and EfficientNet, the recognition accuracy of SimAM-EfficientNetB0 is the highest, reaching 99.47%. It has only 3.83 M parameters and takes an average of 1.08 seconds to test a single image, slightly better than DenseNet121 and other models. Ten thousand adversarial examples generated by GP-MI-FGSM were added to the training set. After pretraining and adversarial training, the accuracy of SimAM-EfficientNetB0 reached 99.78%;
2. The GP-MI-FGSM adversarial attack algorithm has two advantages in the experiment. Its attack success rate is higher than FGSM, I-FGSM, and MI-FGSM, and the perturbation is too small to be detected by human eyes.

Presently, great progress has been made to classify plant diseases, but less attention has been paid to adversarial examples. Therefore, the next step is to focus on adversarial training and other adversarial defense algorithms to enhance the robustness of models.

**Author Contributions:** Conceptualization, H.Y., Y.L. and H.T.; methodology, H.Y., Y.L. and H.T.; formal analysis, H.Y. and Y.L.; investigation, H.Y. and Y.L.; data curation, H.Y. and H.T.; writing—original draft preparation, H.Y. and H.T.; writing—review and editing, H.Y. and H.T.; supervision H.Y., Y.L. and H.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Crr, C.; Psa, A.; Mea, A.; Fn, B. Identification and recognition of rice diseases and pests using convolutional neural networks-sciencedirect. *Biosyst. Eng.* **2020**, *194*, 112–120.
2. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the International Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
3. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
4. Huang, G.; Liu, Z.; Weinberger, K.Q. Densely connected Cconvolutional networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
5. Durmus, H.; Gunes, E.O.; Kirci, M. A hybrid approach for noise reduction-based optimal classifier using genetic algorithm: A case study in plant disease prediction. In Proceedings of the 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, USA, 7–10 August 2017; pp. 1–5.
6. Iandola, F.N.; Moskewicz, M.W.; Ashraf, K.; Han, S.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv* **2016**, arXiv:1602.07360.
7. Zhang, S.W.; Kong, W.W.; Wang, Z. Plant classification method based on dictionary learning with sparse representation. *Acta Agric. Zhejiangensis* **2017**, *29*, 338–344.
8. Guadarrama, L.; Paredes, C.; Mercado, O. Plant Disease Diagnosis in the Visible Spectrum. *Appl. Sci.* **2022**, *12*, 2199. [CrossRef]
9. Kaur, M.; Bhatia, R. Development of an improved tomato leaf disease detection and classification method. In Proceedings of the 2019 IEEE Conference on Information and Communication Technology (CICT), Jeju, Republic of Korea, 6–18 October 2019; pp. 1–5.
10. Tang, W.; Huang, Z. Lightweight model of tomato leaf diseases identification based on knowledge distillation. *Jiangsu J. Agric. Sci.* **2021**, *37*, 9.
11. Yang, L.; Zhang, R.Y.; Li, L.; Xie, X. Simam: A simple, parameter-free attention module for convolutional neural networks. In Proceedings of the International Conference on Machine Learning PMLR, Virtual, 18–24 July 2021; pp. 11863–11874.
12. Tan, M.; Le, Q.V. EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
13. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2019**, arXiv:1412.6572.
14. Moosavi-Dezfooli, S.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582.
15. Papernot, N.; Mcdaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, Germany, 21–24 March 2016; pp. 372–387.
16. Carlini, N.; Wagner, D.A. Towards evaluating the robustness of neural networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 39–57.
17. Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; Abbeel, P. Adversarial attacks on neural network policies. *arXiv* **2017**, arXiv:1702.02284.
18. Zhou, D. Universality of Deep Convolutional Neural Networks. *arXiv* **2020**, arXiv:1805.10769.
19. O'Shea, K.; Nash, R. An introduction to convolutional neural networks. *arXiv* **2015**, arXiv:1511.08458.
20. Golhani, K.; Balasundram, S.K.; Vadamalai, G.; Pradhan, B. A review of neural networks in plant disease detection using hyperspectral data. *Inf. Process. Agric.* **2018**, *5*, 354–371. [CrossRef]
21. Toda, Y.; Okura, F. How convolutional neural networks diagnose plant disease. *Plant Phenomics* **2019**, *2019*, 9237136. [CrossRef] [PubMed]
22. Chen, W.; Chen, J.; Duan, R.; Fang, Y.; Ruan, Q.; Zhang, D. MS-DNet: A mobile neural network for plant disease identification. *Comput. Electron. Agric.* **2022**, *199*, 107175. [CrossRef]
23. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *2*, 2011–2023. [CrossRef] [PubMed]
24. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on 414 Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
25. Guo, C.; Gardner, J.; You, Y.; Wilson, A.G.; Weinberger, K. Simple black-box adversarial attacks. In Proceedings of the International Conference on Machine Learning, Zhuhai, China, 22–24 February 2019; pp. 2484–2493.
26. Croce, F.; Hein, M. Sparse and imperceivable adversarial attacks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4724–4732.
27. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.

28. Robbins, H.E. A Stochastic Approximation Method. *Ann. Math. Stat.* **2007**, *22*, 400–407. [CrossRef]
29. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792.