International Journal of Internet, Broadcasting and Communication Vol.15 No.2 133-143 (2023)
http://dx.doi.org/10.7236/IJIBC.2023.15.2.133

IJIBC 23-2-15

A Model of Strawberry Pest Recognition using Artificial Intelligence Learning

Guangzhi Zhao

Ph.D. Candidate, Dept. of Computer Science and Engineering, Jeonbuk National University
frankzgz1234@gmail.com

Abstract

In this study, we propose a big data set of strawberry pests collected directly for diagnosis model learning and an automatic pest diagnosis model architecture based on deep learning. First, a big data set related to strawberry pests, which did not exist anywhere before, was directly collected from the web. A total of more than 12,000 image data was directly collected and classified, and this data was used to train a deep learning model. Second, the deep-learning-based automatic pest diagnosis module is a module that classifies what kind of pest or disease corresponds to when a user inputs a desired picture. In particular, we propose a model architecture that can optimally classify pests based on a convolutional neural network among deep learning models. Through this, farmers can easily identify diseases and pests without professional knowledge, and can respond quickly accordingly.

Keywords: *Convolutional Neural Network, Strawberry Pests, Diagnosis Model Learning, Deep-Learning, Big Data*

1. Introduction

Nowadays, artificial intelligence, especially the popularity of deep learning algorithms, is remarkable. The reason why deep learning algorithms are popular is that they learn how to perform various tasks based on data and demonstrate performance far superior to humans when given a large amount of data. Among them, Convolutional Neural Network (CNN) was developed by imitating the human visual system and shows very high performance in image processing. However, to achieve high-level performance with a deep learning model, it is necessary to have very good quality data, and it is necessary to design the best deep learning model architecture that can learn the corresponding data well. Table 1 shows List of collected disease and pest data.

Manuscript Received: March. 16, 2023 / Revised: March. 18, 2023 / Accepted: March. 23, 2023
Corresponding Author: frankzgz1234@gmail.com
Tel:*** - **** - ****
Ph.D. Candidate, Dept. of Computer Science and Engineering, Jeonbuk National University, Korea

Table 1. List of collected disease and pest data

Diseases	Pests
Anthracnose leaf scorch Leaf Blight Powdery mildew Gray Mold Disease	two-spotted spider mite greenhouse whitefly Cotton leafworm garen thrips, flower thrips Spiky Slug
leaf spot	Potato aphid, Tomato aphid

In section 2, we will discuss the learning techniques using deep learning models. In section 3, we will present the experimental details. Finally, in section 4, we will conclude our study.

2. Diagnosis Model using Deep Learning

After collecting all the data, we designed a deep learning architecture that can classify the disease and pest data most effectively. The baseline network is based on DenseNet, which was presented at CVPR 2017 [1]. DenseNet is one of the CNN architectures and has shown very high performance [2]. The DenseNet architecture is shown in Figure 1.

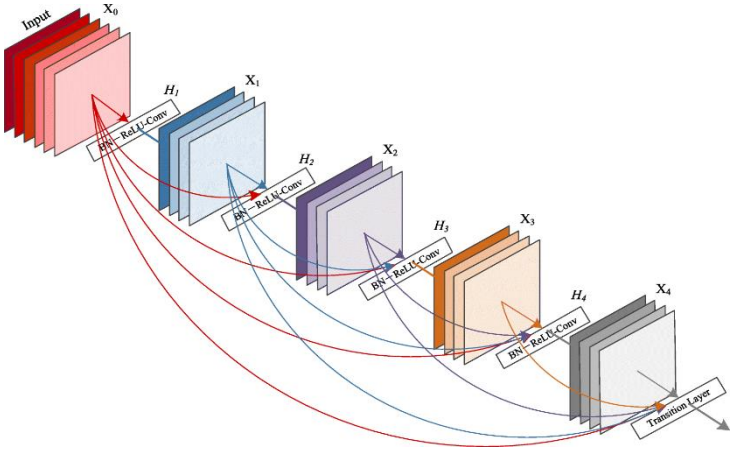


Figure 1. DenseNet architecture

DenseNet is a type of Deep Residual Network that concatenates all output feature maps from the input layer to the output layer and reuses the feature maps, resulting in high performance. Like an ensemble of several shallow networks, Deep Residual Networks that reuse previous output layer feature maps work well even when certain layers' kernels are degraded and collapsed, because they have a very long path for gradients to flow through. This makes them work very well even when designing deep networks [3].

DenseNet improved upon ResNet, which was introduced in 2015 and is the predecessor of Deep Residual Networks [4]. ResNet allows for continuous reuse of feature maps by performing elementwise addition between the previous layer's feature maps and the next layer's feature maps. The operation performed at each layer can be expressed as the following equation (1), and after passing through multiple output layers, the final result is obtained as shown in equation (2). (Here, h represents the output of the previous layer, w is the weight being multiplied, and f represents the activation function.)

$$h_{t+1} = f(h_t * w_{t+1}) + h_t \quad (1)$$

$$h_{t+1} = \sum_{i=0}^{t-1} f_i(h_i * w_{i+1}) + h_t \quad (2)$$

However, there are drawbacks to this Addition method. First, as the feature maps are added through multiple layers, the previous layer's information becomes less clear and fades away. Second, addition can only be performed when the number of channels is the same. To address these issues, DenseNet uses Concatenation instead of Addition. With Concatenation, the information from all layers is stacked Channelwise, so the information is preserved even as it passes through the layers, and output feature maps can be stacked even when the number of channels is different. At each layer, an operation such as (3) is performed, and after passing through multiple output layers, a final result such as (4) is obtained. (h denotes the output of the previous layer, w denotes the weights multiplied, f denotes the activation function, and Z denotes the Concatenation function.)

$$h_{t+1} = Z(f(h_t * w_{t+1}), h_t) \quad (3)$$

$$h_{t+1} = Z(Z_i(f_i(h_t * w_{t+1})), h_t) \quad (4)$$

Furthermore, DenseNet not only reuses the feature maps from the immediately preceding layer, but also utilizes feature maps from all previous layers, providing more paths for gradients to flow through. This leads to higher accuracy, but it requires significant computing resources due to frequent concatenation. To address this issue, this patent proposes a Weakly DenseNet model architecture that reduces unnecessary connections in DenseNet. Residual Connection in Various Models are shown in Figure 2.

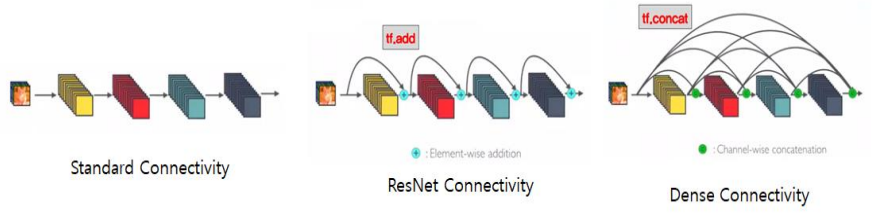


Figure 2. Residual Connection in Various Models

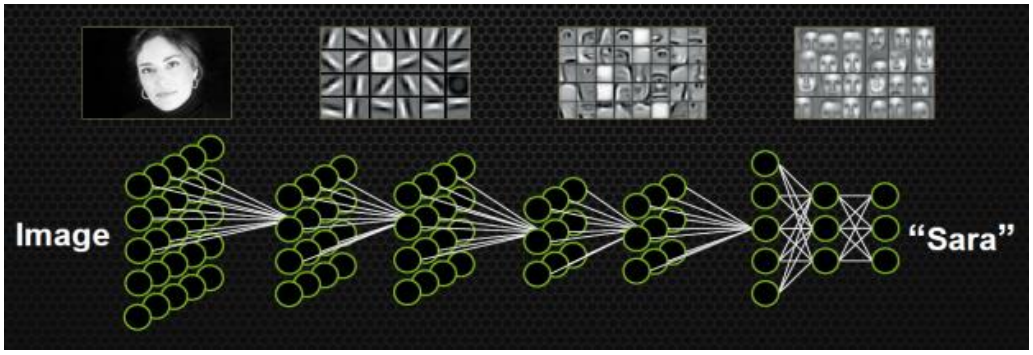


Figure 3. Distribution of feature maps in CNN

As shown in Figure 3, the initial layers of a CNN capture general features such as lines, edges, and surfaces, while middle layers capture more specific features such as eyes, noses, and mouths, and the final output layer captures the most specific features such as the entire human face. This reveals a problem with DenseNet: the feature maps formed in the very early layers of the model provide little help in classifying who the person is, as they capture very general features like lines and edges [5]. However, DenseNet continues to pass all these feature maps to the later layers, consuming a large amount of computing resources in the process.

Therefore, Weakly DenseNet only propagates the very general feature maps from the initial layers to the next layer, similar to ResNet's Skip Connection, without passing them to the classification layer (last layer). By minimizing the inefficient Dense Connection of DenseNet and maximizing the benefits of concatenation, and keeping the channel number from increasing excessively, it is possible to achieve high performance with minimal computing resources. Weakly DenseNet Model Architecture is shown in Figure 4.

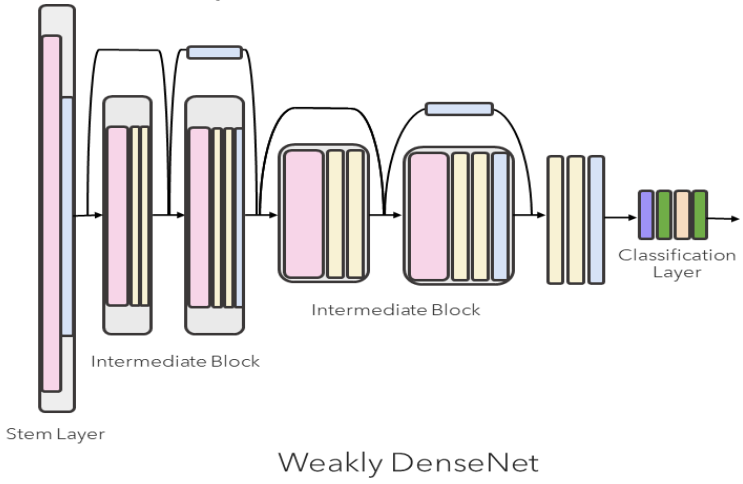


Figure 4. Weakly DenseNet Model Architecture

Figure 5 shows Weakly DenseNet consists of a Stem Layer, Intermediate Blocks, and a Classification Layer.

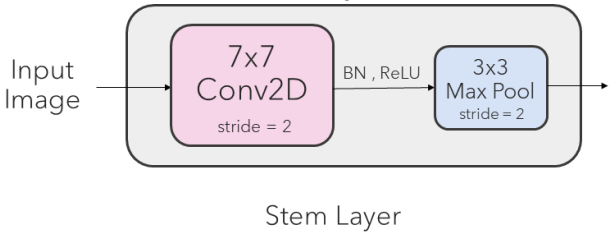


Figure 5. Weakly DenseNet Stem Layer

The Stem Layer is the first block in the network that compresses the input image. It performs a 7x7 convolution operation with a stride of 2, followed by normalization and activation functions, and then a 3x3 max pooling operation with a stride of 2. This reduces the size of the 224x224 image to 56x56. Since it is difficult to perform operations on large-sized images directly, the Stem Layer helps to reduce the image size. Weakly DenseNet Intermediate Building Block (Normal) shows in Figure 6.

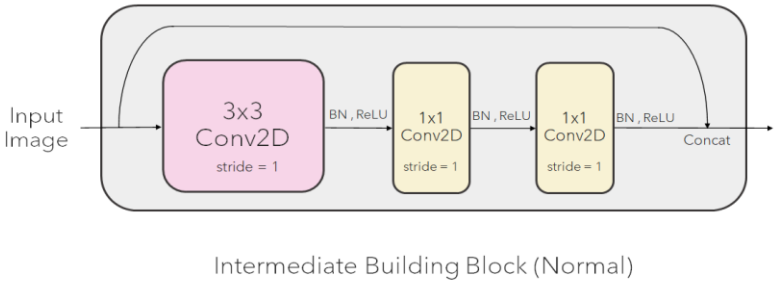


Figure 6. Weakly DenseNet Intermediate Building Block (Normal)

Weakly DenseNet has two intermediate building blocks, where 3x3 convolution and two 1x1 convolution operations are performed. The two 1x1 convolution operations are equivalent to an attention mechanism that strengthens important information. This operates similarly to the Residual Attention Network introduced in 2017, but with lower computing resources while still demonstrating good performance [6]. Attention Mechanism of Residual Attention Network shows in Figure 7.

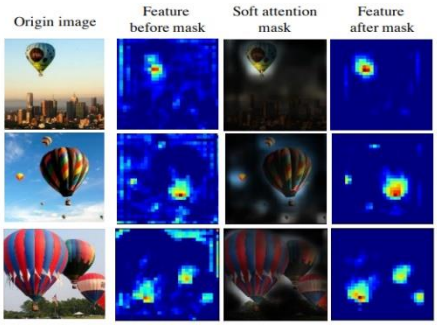


Figure 7. Attention Mechanism of Residual Attention Network

3. Experiments

The two 1x1 Convolution operations included in Weakly DenseNet are as follows where z represents the feature map before the activation function, r represents the feature map after the activation function, K represents the kernel, and x represents the input. The normalization process is omitted for simplicity [7].

$$z^{(1)} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w} K_{ij}^{(1)} * x_{i:i+s,j:j+s} \tag{5}$$

$$r^{(1)} = \max\left(0, z^{(1)}\right) \tag{6}$$

$$z^{(2)} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w} K_{ij}^{(2)} * r_{i:i+s,j:j+s}^{(1)} \tag{7}$$

$$r^{(2)} = \max\left(0, z^{(2)}\right) \tag{8}$$

A 1x1 convolution operation multiplies the input feature map by a convolution kernel of size 1. This convolution kernel is learned through the backpropagation process and can highlight important features, while masking unimportant ones (negative pixels) through the ReLU activation function, preventing their values from being inverted.

$$\frac{\partial Loss}{\partial K^{(2)}} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w} \frac{\partial Loss}{\partial r^{(2)}} * \frac{\partial r^{(2)}}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial K^{(2)}} \tag{9}$$

$$\frac{\partial Loss}{\partial K^{(1)}} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w} \frac{\partial Loss}{\partial r^{(2)}} * \frac{\partial r^{(2)}}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial r^{(1)}} * \frac{\partial r^{(1)}}{\partial z^{(1)}} * \frac{\partial z^{(1)}}{\partial K^{(1)}} \tag{10}$$

$$K^{(2)} := K^{(2)} - \eta \frac{\partial Loss}{\partial K^{(2)}} \tag{11}$$

$$K^{(1)} := K^{(1)} - \eta \frac{\partial Loss}{\partial K^{(1)}} \tag{12}$$

The gradients used in training are computed as follows. K (2) refers to the second 1x1 Convolution kernel and K (1) refers to the first 1x1 Convolution kernel. The gradients of the two vectors K (1)(2) are proportional to the input they will be multiplied with, so the values are linearly adjusted as the input values increase.

$$\frac{\partial Loss}{\partial K^{(2)}} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w} \frac{\partial Loss}{\partial Out} * 1 * r^{(1)} \tag{13}$$

$$\frac{\partial Loss}{\partial K^{(1)}} = \Sigma_{i=0}^{h}\Sigma_{j=0}^{w}\frac{\partial Loss}{\partial Out} * 1 * K^{(2)} * 1 * x \quad (14)$$

(K (2) is Must bigger than 0 by ReLU Activation)

To put it more intuitively, if the pixel value of an important part of an image with a pest is 8.2, and the pixel value of an unimportant part without a pest is 0.2, performing a 1x1 convolution will linearly adjust the values. If K is 4, the pixel values of the pest areas will be adjusted to 32.8, and the pixel values of the non-pest areas will be adjusted to 0.8, which is a more distinct difference than the previous difference of 8 between the two positions. As a result, important and unimportant parts can be better separated, and even a tiny difference, such as 1.2 between the legs and the body, can be more accurately distinguished after multiplying by K=4. Figure 8 shows 1x1 Convolution structure.

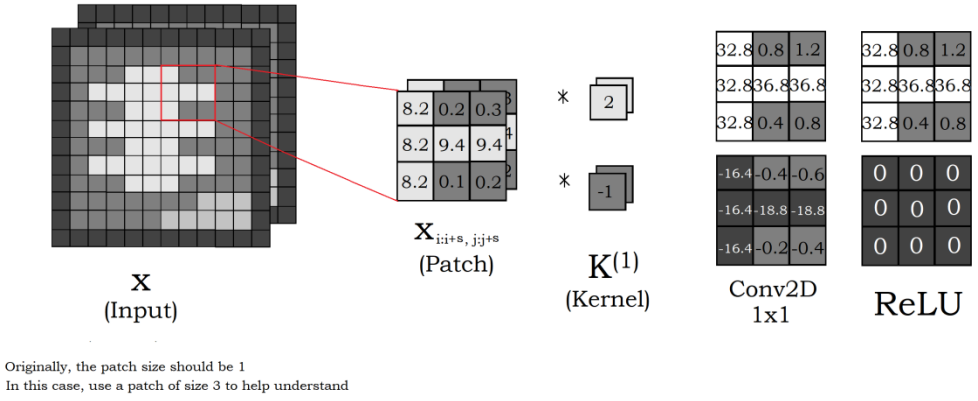


Figure 8. 1x1 Convolution

If K were negative, all pixel values would become negative, with important areas experiencing a sharp drop in pixel values and unimportant areas experiencing a slight decrease. (If K were -3, 10 would become -30 and 0.1 would become -0.3.) Such a feature map inverts information, making classification more difficult. Therefore, this feature map becomes a residual channel with all pixels being 0 after passing through the ReLU activation function.

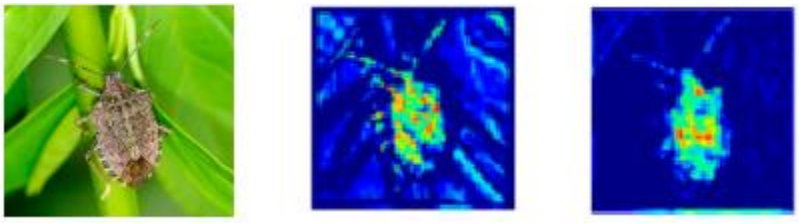


Figure 9. Attention Mechanism of Weakly DenseNet

Figure 9 shows the image output from the actual model. It can be seen that the Attention Mechanism is functioning properly. The first image is the input image, and the second image is the image after passing through a 3x3 Convolution operation. When a 1x1 Convolution is applied twice, the pixel values of the pest in the center are further enhanced, while the pixel values of the surrounding leaves and branches become very faint. What the model wants is the appearance of the pest, not the appearance of the leaves. Through this Attention process, the model can better classify the pest. Subsequently, the Concatenation operation is performed with the original input. By performing Concatenation, Degradation (a phenomenon in which all gradients become zero and can no longer flow backward during Backward Propagation. In the case of ReLU, only nodes that were positive in the previous layer are multiplied by 1, and nodes that were negative are

multiplied by 0, but if all nodes are 0, all gradients that flow in become 0, and the gradient can no longer flow) can be prevented to some extent. Many paths are created for the gradient to flow, so even if a certain K is wrongly learned and converges to a negative value, the gradient can still flow without problems toward the previous layer.

$$h_{t+1} = Z(f(h_t * w_{t+1}), h_t) \tag{15}$$

As shown in the figure 10, the left network without connections cannot function properly if the f2 layer degrades, but when connections are added as in the right figure, many paths for gradient flow are created. By connecting each layer with connections, the network operates as if it were 8 shallow networks as shown in the figure, making it possible to create a model that is very robust to the degradation problem.

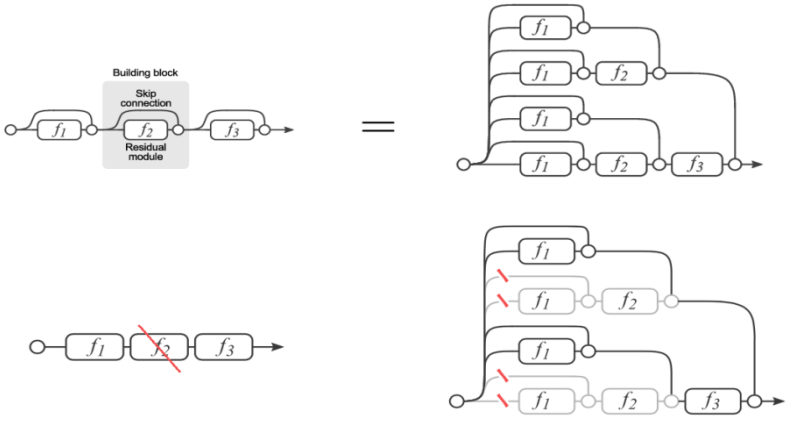


Figure 10. Concatenation Connection

Then, the image size is reduced by the Reduction Block. Weakly DenseNet Intermediate Building Block (Reduction) shows in Figure 11. The Reduction Block is generally similar to the Normal Block mentioned above, but performs MaxPooling operation on the image to reduce the image size by half. Weakly DenseNet Classification Layer shows in Figure 12.

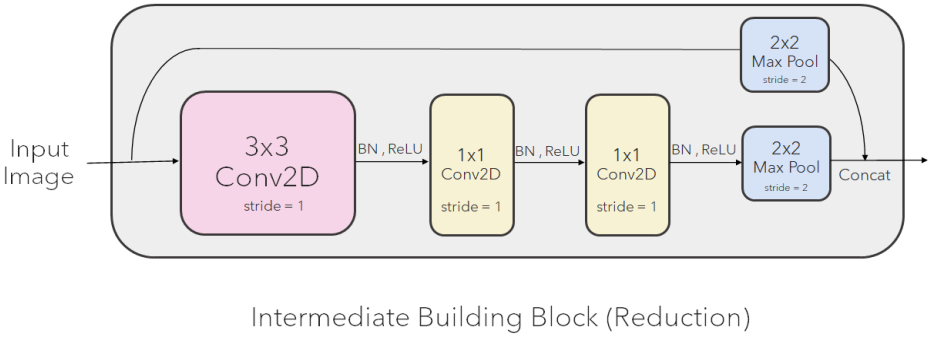


Figure 11. Weakly DenseNet Intermediate Building Block (Reduction)

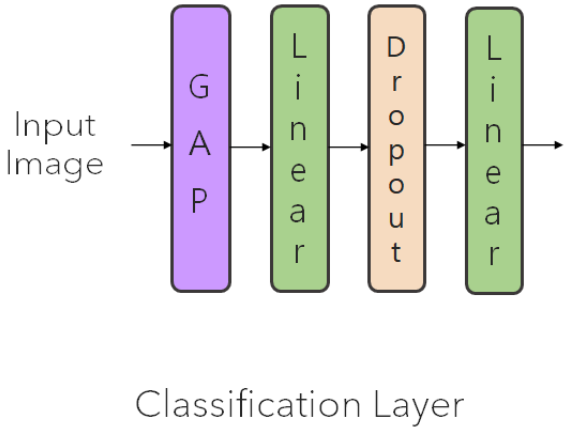


Figure 12. Weakly DenseNet Classification Layer

Table 2. Overall model organization

Block	Output Size
Initial Block (a)	56 x 56 x 32
Intermediate Block (b)	56 X 56 X 96
Intermediate Block (c)	28 X 28 X 192
Intermediate Block (b)	28 x 28 x 384
Intermediate Block (c)	14 X 14 x 768
1 x 1 conv, stride 1	14 X 14 x 512
1 X 1 conv, stride 1	14 X 14 x 512
2 x 2 max pool, stride 2	7 x 7 x 512
Classification Block (d)	1 X 1 X 24

The table 2 illustrates the overall architecture of the model, which takes an input image of size 224 x 224 x 3 (RGB channel) and transforms it into a tensor of size 7 x 7 x 512 before going through the Classification Layer for classification. With a total of 16 layers, this deep learning model architecture demonstrates exceptional performance compared to other existing models.

To create a more accurate model, various optimization techniques were applied during training. Data augmentation techniques were used to increase the amount of data by rotating, shifting horizontally and vertically, shearing, zooming, and horizontally flipping the available images. Figure 13 shows Augmentation techniques for model training.

Operation	Value
Rotation	$[0^{\circ}, 15^{\circ}]$
Width shift	[0, 0.2]
Height shift	[0, 0.2]
Shear	[0, 0.2]
Zoom	[0.8, 1.2]
Horizontal flip	-

Figure 13. Augmentation techniques for model training

Algorithm 1. Learning Rate Schedule

Input: Patience P, decay θ, validation loss L
Output: Learning rate γ
Initialize $L = L_0, \gamma = \gamma^0$
$i \leftarrow 0$
while $i < P$ **do**
  if $L \leq L_i$ **then**
    $i = i + 1$
  else
    $L = L_i$
    $i = i + 1$
  end if
end while
if $L = L_0$ **then**
  $\gamma = \gamma * \theta$
end if

To enable the model to reach the global minimum instead of a local minimum, a learning rate scheduling algorithm was designed. The learning rate is reduced by a factor of θ (decay rate) every P (patience) epochs. P and θ are hyperparameters, and for this model, P was set to 5 and θ was set to 0.8. Additionally, to reach the global minimum, Nesterov Momentum was set to 0.9, and the initial learning rate was set to 0.01[8].

4. Result

The table below Table 3 and table 4 shows the experimental results. A total of 8 models and control experiments were conducted. All models used the same dataset, data augmentation techniques, and learning rate scheduling.

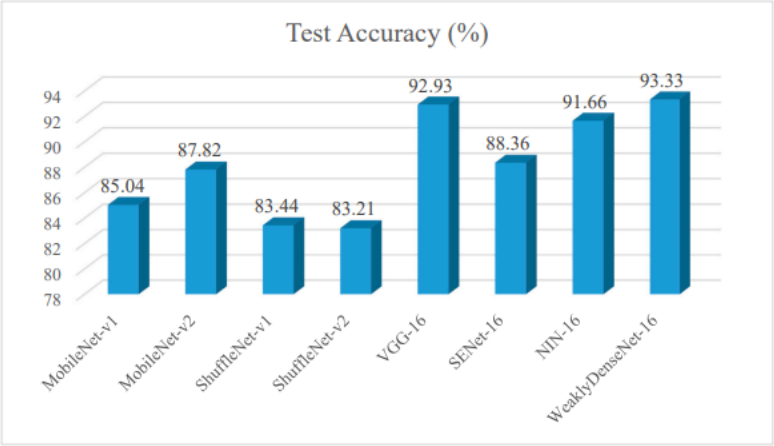
Table 3. result (compare)

Model Name	Training Accuracy	Validation Accuracy	Model Size (MB)	Training Time (ms)/Batch Size
MobileNet-v1	99.23	85.45	25	152
MobileNet-v2	99.28	87.97	33.9	198
ShuffleNet-v1	99.13	83.58	28.8	145
ShuffleNet-v2	98.72	83.58	42	144
VGG-16	99.82	93	120.2	303
SENet-16	99.10	88.71	19.5	138
NIN-16	99.63	91.84	19.6	137
WeaklyDenseNet-16	99.83	93.42	30.5	138

'NIN' represents Network in Network.

Weakly DenseNet is a lightweight version of DenseNet that reduces the unnecessary densely connected la yers, greatly decreasing the number of operations (FLOPs) required for the model. In each layer, it mainly pe rforms 1x1 convolution operations rather than 3x3 convolutions, resulting in a very small number of paramet ers. Therefore, it has a similar model size to other lightweight models such as MobileNet and ShuffleNet [9,10].

Table 4. result (Bar graph)



When tested on the test data (unseen data), Weakly DenseNet showed better performance (92.93% < 93.42%) than VGG-16 , which has a huge amount of parameters close to 120MB, and outperformed SENet, which is a similar attention-based network [11,12]. In other words, Weakly DenseNet has a lightweight model size, but shows much better performance than other giant networks. Network In Network, which is a network that performs 1x1 convolution after 3x3 convolution similarly to Weakly DenseNet, also showed good performance with a small size [13]. Therefore, it can be seen that performing 1x1 convolution after 3x3 convolution is more effective in accurate classification of plant diseases and insect pests, and adding connections by performing concatenation operation can further improve the performance (91.84% < 93.33%) [14].

5. Conclusion

In conclusion, this study proposes a big data set of strawberry pests and an automatic pest diagnosis model architecture based on deep learning. By directly collecting and classifying over 12,000 image data related to strawberry pests, we were able to train a deep learning model and develop an automatic pest diagnosis module. This module enables farmers to easily identify diseases and pests without professional knowledge and respond quickly accordingly. The effectiveness of the proposed model architecture, which optimally classifies pests based on a convolutional neural network, was demonstrated through experiments. Overall, this study highlights the potential of deep learning algorithms in agriculture and provides a practical solution for pest diagnosis in strawberry farming. Further research can be conducted to expand the application of deep learning algorithms in various agricultural domains.

Acknowledgement

This work was supported by project for Joint Demand Technology R&D of Regional SMEs funded by Korea Ministry of SMEs and Startups in 2023.(Project No. RS-2023-00207672)

References

[1] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[2] B. J. Jeong, M. S. Kang, and Y. G. Jung, "A Study on the Facial Expression Recognition using Deep Learning Technique, " International Journal of Advanced Culture Technology, vol. 6, no. 1, pp. 60–67, Mar. 2018. https://doi.org/10.17703/IJACT.2018.6.1.60

[3] Veit, Andreas, Michael J. Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow

networks." Advances in neural information processing systems 29, 2016.
[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[5] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." Advances in neural information processing systems 27, 2014.
[6] Wang, Fei, et al. "Residual attention network for image classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[7] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
[8] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning. PMLR, 2013.
[9] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861, 2017.
https://doi.org/10.48550/arXiv.1704.04861
[10] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[11] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556, 2014.
https://doi.org/10.48550/arXiv.1409.1556
[12] Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[13] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).
https://doi.org/10.48550/arXiv.1312.4400
[14] Malrey Lee, "Final report of Industry-academia joint technology development project. " Jan.2019