PiTLiD: Identification of Plant Disease From Leaf Images Based on Convolutional Neural Network

Kangchen Liu and Xiujun Zhang

Abstract—With the development of plant phenomics, the identification of plant diseases from leaf images has become an effective and economic approach in plant disease science. Among the methods of plant diseases identification, the convolutional neural network (CNN) is the most popular one for its superior performance. However, CNN's representation power is still a challenge in dealing with small datasets, which greatly affects its popularization. In this work, we propose a new method, namely PiTLiD, based on pretrained Inception-V3 convolutional neural network and transfer learning to identify plant leaf diseases from phenotype data of plant leaf with small sample size. To evaluate the robustness of the proposed method, the experiments on several datasets with small-scale samples were implemented. The results show that PiTLiD performs better than compared methods. This study provides a plant disease identification tool based on a deep learning algorithm for plant phenomics. All the source data and code are accessible at https://github.com/zhanglab-wbgcas/PiTLiD.

Index Terms—Plant phenomics, plant disease, leaf image, convolutional neural network, deep learning

1 Introduction

Plant phenotype refers to plant phenotypic traits for reflecting characteristics of crops such as the yield, quality, stress resistance and so on [1]. With the rapid development in plant phenotype, the high-throughput phenotyping technology has been studied in plant science for plant diseases estimation and classification [2]. Plant phenotyping methods can detect and classify diseased plants in time to avoid economic losses thanks to the advance in sensing and data analytic technologies [3]. Due to unidentified plant diseases may cause huge losses, e.g., cucumber is severely affected by more than 30 diseases during its growth cycle; fungal pathogens can cause yield losses up to 50% in winter wheat; many living or biotic substances affect the fruit of the plant tomatoes and other plant parts like roots, stems and plant's leaves [4], [5], [6]. Phenotypic analysis can help farmers timely identify the real types of diseases, then take corresponding preventive measures and precisely target the amount of fertilizer or pesticide respectively as required to improve the quality and yield of crops so as to reduce the economic loss of farmers [7]. Thus, phenotypic analysis is playing a significant role in many fields like botany, agronomy, and so on [8], [9].

Current phenotypic analysis methods for plant diseases using handheld tools or visual observation with naked eyes were laborious, expensive, time-consuming and experience-dependent for the expert's judgment [10], [11]. As a breakthrough in plant phenomics, the emergence of deep learning has made tremendous progress in complex phenotyping tasks [12], [13], [14], [15]. As a state-of-the-art deep learning approach, the convolutional neural network (CNN) has been proven to perform excellently in various aspects, such as face recognition, objection detection, machine translation, text classification, and cancer therapy [16], [17], [18], [19], 20]. In recent years, the agricultural Internet of things (IoTs) has become one vital infrastructure for timely access to data in the field. In modern precision agriculture, it is feasible to acquire and identify images of plant diseases online in real time and accurately through the IoTs system [4], [21]. For the performances of CNN-based methods were nearly close to human-level in computer vision tasks, a wide range of agricultural applications based on CNNs have been produced, such as plant disease diagnosis and detection, assessment of plant disease severity, plant organ detection and counting, and weed recognition [22], [23], [24], [25], [26], [27], [28], [29], [30]. These agricultural applications increase farm income while reducing labor costs extremely. Among these agricultural problems, the successful classification of plant diseases can help to improve the quality and/or quantity of agricultural products, reduce the misuse of chemical sprayers such as fungicide/herbicide and protect the environment at the same time [31].

Although current deep learning approaches have made great success, many researchers recognize that the insufficiency of training samples affects the performance of CNNs seriously [32]. When training the dataset with small sample size, the models are susceptible to over-fitting, which

- *Kangchen Liu is with the Key Laboratory of Plant Germplasm Enhancement and Specialty Agriculture, Center of Economic Botany, Core Botanical Gardens, Wuhan Botanical Garden, Chinese Academy of Sciences, Wuhan, Hubei 430074, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: liukangchen18@mails.ucas.ac.cn.*
- *Xiujun Zhang is with the Key Laboratory of Plant Germplasm Enhancement and Specialty Agriculture, Center of Economic Botany, Core Botanical Gardens, Wuhan Botanical Garden, Chinese Academy of Sciences, Wuhan, Hubei 430074, China. E-mail: zhangxj@wbgcas.cn.*

Manuscript received 18 December 2021; revised 2 June 2021; accepted 26 July 2022. Date of publication 1 August 2022; date of current version 3 April 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 32070682 and 61402457, in part by Technology Innovation Zone Project under Grants 1716315XJ00200303 and 1816315XJ00100216, and in part by CAS Pioneer Hundred Talents Program.
(Corresponding author: Xiujun Zhang.)
Digital Object Identifier no. 10.1109/TCBB.2022.3195291

damages the performance of model and decrease classification accuracy dramatically for the classification of plant images [22]. The ideal state of a large sample dataset in the actual work is not easy to achieve for the influence of experimental environment and device. In this case, it is necessary to develop robust and feasible methods suitable for the dataset with small sample size. To resolve the small sample problem in data analysis, some methods have been proposed. The SSF-CNN method used a dictionary-based filter learning algorithm to initial the filter's structure to address small sample size limitation [33]. The deep convolutional neural network (DCNN) method used transfer learning to solve small sample problem with a global average pooling instead of the fully connected layers. This method decreases the number of parameters and improves the performance [34]. The Layer-Sequential Unit-Variance (LSUV) method was proposed to initialize the convolutional layer's weights and normalize it to the unit variance [35], [36]. A relatively compact convolutional neural network (CVL17) was specially designed for small and imbalanced datasets in gestational age estimation. The compact neural network can reduce complexity of deep convolutional neural network and mitigate over-fitting problem [37].

In this work, we proposed a new deep learning approach, namely PiTLiD, based on pretrained Inception-V3 convolutional neural network and transfer learning to identify plant leaf disease from phenotype data with small sample size. We demonstrated the effectiveness and feasibility of PiTLiD in phenotyping task of diseases classification. PiTLiD eliminated effects of poor samples on image recognition and classification and got an overall accuracy of $99.45 \pm 0.17\%$, which is the highest prediction accuracy in several phenotyping approaches. PiTLiD also showed good performance on other small disease datasets. This study provides a plant disease identification tool based on deep learning algorithm for plant phenomics. It is hoped that PiTLiD would accelerate the study of phenomics. All the source data and code are accessible at https://github.com/zhanglab-wbgcas/PiTLiD.

2 Methods

2.1 Data Augmentation

In our work, apple leaf diseases dataset only includes 120 training images over 4 classes. Note that there are a small number of samples, data augmentation is an important part of apple diseases classification, it can enlarge the training data amount through random transformations then improve the generalization ability of the model and prevent over-fitting of the network caused by insufficient dataset. The methods of data augmentation in deep learning mainly include image scaling, cropping, flipping, shifting, rotation, and zoom [38], [39], [40], [41]. More complex techniques can include adding noise, pose and lighting, geometric transformations, radial transform or image compression [42], [43], [44]. Too many regularization operations of the image were eliminated to retain more original image information. The augmentations used to create the augmented or synthesized images were as follows: (1) random flip, images are flipped horizontally and vertically; (2) random rotate, images are randomly rotated within the range of [0,90] degrees; (3) random shift, images are randomly shifted horizontally and vertically; (4) random zoom enlarges or shrinks images in a certain proportion; (5) random shear keeps x-coordinate (or y-coordinate) of all the pixel points unchanged, while the y-coordinate (or x-coordinate) is shifted proportionally.

2.2 Convolutional Neural Network

The convolutional neural network (CNN) was first introduced by Fukushima in 1980, which is a feed-forward neural network that includes several convolution operations. Based on its special features of shared-weight and local connectivity, data processing and feature learning capabilities of CNN are excellent [45], [46]. Weight sharing means that all spatial locations of the picture use the same convolution kernel, and when the kernel is moving, the weight in it will not be changed, which further effectively reduces the number of parameters needed for a convolution layer and eases computational burden [47]. Based on local connectivity, each neuron in a convolutional layer is only connected to a part of the input neurons to get local information, then all of local information gathering to obtain the global information [48]. Given these advantages, CNN is widely used in the field of crop disease recognition and has achieved remarkable achievements.

The architecture of CNN is mainly composed of data input layer, convolutional layer, pooling layer, fully connected layers and output layer, as shown in Fig. 1 [49], [50]. The input layer is the entrance of the data set to be identified. In order not to restrict the performance of the network, we would better ensure that the size of the input data is as close as possible to the network requires to facilitate convolution operation.

The convolutional layer plays a vital role to perform the feature extraction on the image by a series of linear or nonlinear operations. It comprises a set of filters (also called kernels). The filter length should be smaller than the input image length commonly. Fig. 1**A** presents how each filter detects and extracts the local information. Different filters in convolutional layers can learn different image features. The output of this layer is called a feature map, which is depicted in Fig. 1**B**. The compute form of the convolution process is defined as:

$$X_j^{\mathrm{l}} = f\left(\sum\nolimits_{i \in M_j} X_{ij}^{l-1} * K_{ij}^{l} + b_j^{l}\right), \tag{1}$$

where X_j^l is the output of the convolutional operation, it represents the j-th feature map produced by l-th convolutional layer, X_i denotes the i-th input data point of the convolutional layer, and M_j be the number of such data points, K_{ij}^l is the convolution kernel connecting the i-th data point to the j-th feature map, each output feature having a unique bias vector b_j^l , function $f(.)$ is activation function, and "$*$" represents a one-dimensional convolution operation.

After each convolutional layer, it is conventional to apply an activation function to introduce non-linearity in a CNN. In the present works, the rectified linear unit (*ReLU*) is chosen as the activation function and proved to be more effective in avoiding gradient vanishing or explosion than the traditional activation functions like sigmoid and hyperbolic tangent, also improving discriminative performance, which have been highly successful for computer vision tasks.

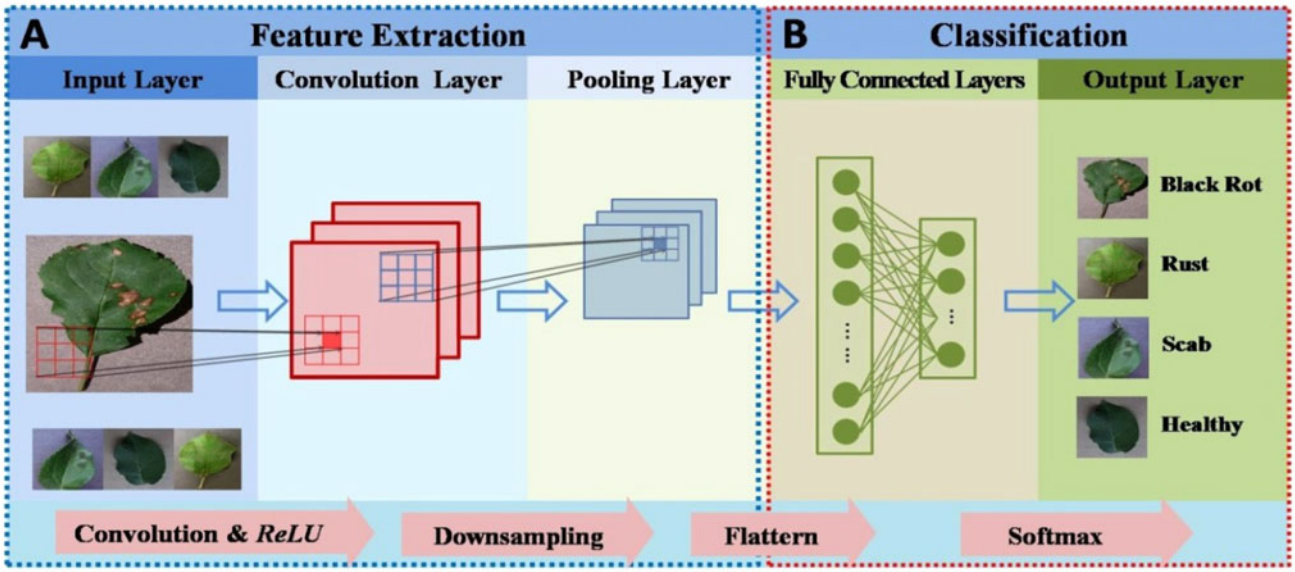


Fig. 1. The basic structure of CNN. **(A)** Feature extraction module. In convolutional layer, the red rectangle in the image is a filter which convolves over the input image and calculates the scalar product between the filter and the input at every position to extract feature. The red grid pointed by the arrow is the result of a filter at current position. At each spatial position, the filter produces a result. Finally, a feature map is outputted after convolution. In pooling layer, the filter window (blue rectangle) is slid over the feature map to perform down-sampling by using a pooling function. Down-sampling can reduce small patches of the feature map into single pixels and retain the key information of input image. The blue grid pointed by the arrow is the result of a filter at current position. (**B)** Classification module. The output (multiple dimensions of feature map) of the pooling layer is converted to one dimension and then passed to fully connected layers. In output layer, the class probability of each class label can be computed by applying the *softmax* function to the data in the fully connected layers, and then the probability values are sent to the output layer for classification result.

ReLU transforms the input as:

$$ReLU(x) = max(0, x). \tag{2}$$

The pooling layer is always placed behind the convolutional layer to perform the down-sampling operation of the feature map with two advantages. On the one hand, it can reduce the dimensionality of the feature map, thus the number of trainable parameters is decreased to low computational complexity of model training, also control over-fitting. On the other hand, it ensures crucial features are not lost [45]. Common pooling methods include max pooling and average pooling, the former calculates the maximum of the output, and the latter calculates mean values. In this paper, we used max pooling, which can enhance the slight signal of the small regions in the images, therefore it is best suited for our diseases dataset that contains a tiny difference between each leaf image [47]. The output of the max pooling layer defined as P can be calculated by:

$$P_n^j = max\left\{Y_n^{j*l_p:(j+1)*l_p}\right\}, \tag{3}$$

where P_n^j is the output value of the j-th feature map from the nth data point. l is the length of sub-region. $max(.)$ denotes the function to compute the max value. As the window moves across j-th feature map, function max chooses the largest value in the window to build a new matrix and pass it to next layer.

The output of the final convolutional or pooling layer represents the feature extracted from the input data. It is flattened to one dimension and passed to the fully connected layers to perform classification or regression. In fully connected layers, like its name, the neurons at adjacent layers are all connected to each other. The fully connected layers play a "classifier" role in CNN when performing classification, each neuron in the last connected layer represents one class of the classification task, the prediction score of each category can be counted by *softmax* function [51]. The operation can be formulated as

$$Y = f(wx + b), \tag{4}$$

where Y is the output, and function $f(.)$ represents the activation functions, w is a weight matrix created by the network layer, b is a bias vector.

Dropout is added in training process to avoid over-fitting problem of model training. The principle of dropout prevents over-fitting is to randomly drop units from the neural network during training [52]. In a dropout layer, part neurons are set to 0 with a probability of 0.5, these dropped neurons do not contribute to the forward-propagation and back-propagation during the training phase, so the phenomenon of over-fitting in model training was decreased.

The architecture of all the neural networks is constructed by minimizing the following cross-entropy loss function between the predicted and true label values [45]. The process is expressed as

$$L = -\frac{1}{n}\sum\nolimits_x \hat{y} * \ln y + \left(1 - \hat{y}\right) * \ln(1 - y), \tag{5}$$

where y is the model output, and$y^{\wedge}$ is the value of the real label.

The gradient descent and back propagation algorithm are used for optimization and minimizing the cost function L [45]. Thus, the mathematical form of updating weight and bias is shown below:

$$w = w - \lambda\frac{\partial_L}{\partial_w},\ b = b - \lambda\frac{\partial_L}{\partial_b}, \tag{6}$$

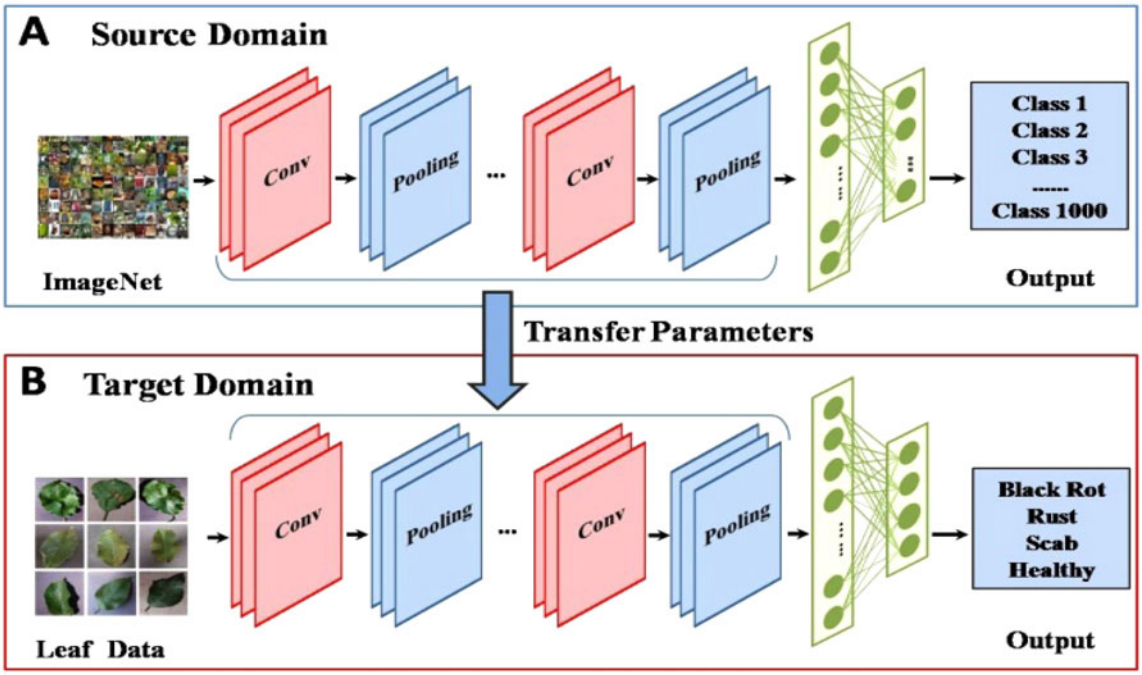


Fig. 2. Illustration of transfer learning. (**A**) The source model in source domain is pretrained on the ImageNet database for classification. (**B**) The target model in target domain is initialized with the weights and parameters from source model. The classification module is replaced by a new one to compensate for the difference between source task and target task in image category and attributes.

where w denotes the weight, b represents the bias, and l is the learning rate.

2.3 Global Average Pooling

CNNs adopt the fully connected layers after convolutional layer or pooling layer, each neuron in fully connected layers is completely interconnected and brings many parameters, then the training time and the demand for computing memory are increased. Besides, fully connected layers can't interpret how each filter contributes to each category of classification because of the black-box effect. To preserve the ability of classification and meanwhile to avoid black-box effect, we determined to use global average pooling (GAP) layer that connects to *softmax* layer directly.

A GAP layer calculates the average of each feature map of the last convolutional layer to generate a feature point for each input image, and then all feature points are concatenated into a vector which is directly input to *softmax* function for completing the classification. Thus, a GAP layer effectively reduces a large number of parameters of the fully connected layers and alleviates the risk of over-fitting, meanwhile the contributions of filters become interpretable [53]. In addition, it makes no restrict to the size of input data, which denotes that we can use it to deal with the input image of any size while the fully connected layers can only deal with one certain dimension [54]. The GAP layer is represented by the formula below:

$$c = \sum\nolimits_j \left(M_j * p\left(F_j\right)\right) \tag{7}$$

where c denotes the total feature contributions. M_j represents the contribution weights. Function $p(.)$ is a pooling operation with kernel size is 1^*1 and F_j denotes the j-th feature in the last convolutional layer.

3 Results

3.1 Framework of PiTLiD Method

3.1.1 Training Process and Measures

We implemented the training process by utilizing a pretrained network which was created on Inception-V3 network and loaded with pretrained weights from ImageNet, the process is illustrated in Fig. 2**A**.

In the first step, the classification module was revised. Fig. 3**A** shows the structure of Inception-V3 network. We abandoned fully connected layers of Inception-V3 model to make the GAP layer connected with *softmax* layer directly, and the output layer was changed from a thousand classes to four for apple diseases classification. Thus, we got a pretrained model which had 51 layers, including 5 complete convolution layers, 3 complete pooling layers, and one *softmax* classifier layer. The remaining 42 layers were composed of 3 inception blocks. Among them, Block 1 contained 3 modules with a total of 9 layers, Block 2 contained 5 modules with a total of 23 layers, and Block 3 contained 3 modules with a total of 10 layers.

In the second step, hyper-parameters were tuned to find the best set. After revised classification module, the input data was provided to the pretrained network and various combinations of hyper-parameters were tested. The hyper-parameters we evaluated mainly were the optimization algorithm, batch size, epochs, steps, learning rate, and so on.

In the third step was transferred the parameters and weights. The parameters and weights of feature extracted module were transferred from source domain to target domain. Due to the category number of the classification tasks in source domain and target domain are different, the classification module in target model was initialized with

random weights so that the image feature of apple diseases images rather than ImageNet images could be learned.

Finally, an attempt was made to determine the best number of fine-tuning layers. Different layers were fine-tuned, their corresponding parameters and weights were updated on the apple disease dataset. The rest layers were frozen and their parameters and weights were fixed during fine-tuning. We attempted three different fine-tuning settings: the first seven layers were frozen and the remaining layers were fine-tuned; the first ten layers were frozen and the rest of layers were fine-tuned; the whole structure of the neural network was fine-tuned (Fig. 2**B**).

3.1.2 Transfer Learning Based on Inception-V3 Network

In some research fields, it is hard to obtain enough available data or create labeled data to train CNN. What's more, training CNN from scratch is always accompanied by heavy computing burdens and much time. Transfer learning, which is a powerful tool to improve generalization and boost robustness of model, has been further developed to solve these challenges. Transfer learning aims to migrate the knowledge of a source domain which uses plenty of labeled source data to a relevant target domain, as illustrated in Fig. 2 [55], [56], [57], [58]. Thus, target model can achieve better performance even if the training data is insufficient. In our work, the pretrained model is trained with a subset of 1000 classes separated from ImageNet. ImageNet is a wide variety of large-scale image database for vision research. It consists of over 15 million labeled high-resolution images in over 22,000 categories [59].

Two common transfer learning strategies in deep learning are deep feature extraction and fine-tuning [57]. In deep feature extraction, the pretrained model served as a feature extractor, the outputs of the last fully connected layers of it are obtained and used as features. After deep feature extraction, features are input to machine learning algorithms for classifying. In fine-tuning, the new small dataset is provided to the pretrained model, some layers of the pretrained network can be fixed while the rest layers can be fine-tuned. The pretrained network is retrained and the parameters and weights of it are updated to learn new properties of the new data. Different classification accuracy can obtain by fine-tuning different layers in feature extracted module, because they contain different characteristics. The features extracted from the low layers are more generic such as edges, tints, shades, and textures, which are applicable to multiple visual tasks. Contrarily, high layers features are more specific for every distinct task [60[, [61]. Traditional transfer learning only fine-tunes the last layers in model. In this work, we retrained different layers in feature extraction module.

The Inception-V3 model was used as the basic model to perform transfer learning, which is a deep convolution neural network model proposed in 2015, consisting of multiple convolution layers, max-pooling layers, inception modules, average pooling layer, fully connected layers, and *softmax* layer [62]. The structure of Inception-V3 model is displayed in Fig. 3A, and the details of each inception module are displayed in Figs. 3B, 3C, 3D, 3E, and 3F. Some calculate principles and optimization ideas make it wider and deeper than earlier networks like VGG and AlexNet, as well as save computational cost and prompt accuracy. $n \times n$ convolution was replaced by a $1 \times n$ convolution followed by an $n \times 1$ convolution, and $1^{*}1$ convolutional kernel is widely used. Several convolutions and maximum pooling are calculated simultaneously and concatenated. Auxiliary classifiers were added to help training results become more steady and gradient converge quicker [63], [64]. In this way, the Inception-V3 model is good performance even under less computational memory with complex computational problems.

3.1.3 Learning Rate Methods

Learning rate (LR) is a vital hyper-parameter when training a neural network, an optimal learning rate could improve the performance of network [65], [66]. There are three common learning rates. The constant learning rate is the simplest learning rate strategy whose value is remained unchanged during training. It's difficult to find a suitable value of constant learning rate because in the early stages of training. If the value is small, the training speed will slow down, and the training is easy to fall into local best and lead to over-fitting. If the value is large, however, the oscillation will occur for the loss function and the global best will be hard to be found at the end of training. We need to repeat numerous experiments to try various values of constant learning rate for finding the best one.

To eliminate this experimental requirement, the decay learning rate would be a good choice whose initial value decreases by a certain rate after pre-defined epochs [67]. The decay learning rate can make the model converge better but converge will be terminated when training encounters the saddle point. The cyclical learning rate (CLR) provides a solution to get out of saddle point.

The value of CLR is changed instead of kept fixed. It varies cyclically within the range of two reasonable boundary values, namely lower bound (base_LR) and upper bound (max_LR) respectively. Overall, rising or falling of learning rate is favorable even if it might be detrimental to network performance temporarily. CLR can help model training overpass saddle points plateaus more quickly by oscillating the value inside a band [68]. Thus, CLR can prompt the convergence speed of the loss function and achieve higher accuracy with fewer epochs without the demand for numerous experiments and tuning simultaneously [69]. As follows are three CLR policies.

1) Triangular 1 policy. This policy varies the value of learning rate linearly between the minimum (base_LR) and the maximum (max_LR).
2) Triangular 2 policy. This policy is the same as the triangular policy except that the maximum learning rate is half (shown in Fig. 3**G**).
3) Exponential range policy. The value of learning rate varies between the minimum and maximum boundaries. The value of max_LR is declined with an exponential factor of gamma in each iteration, while base_LR is kept unchanged [68].

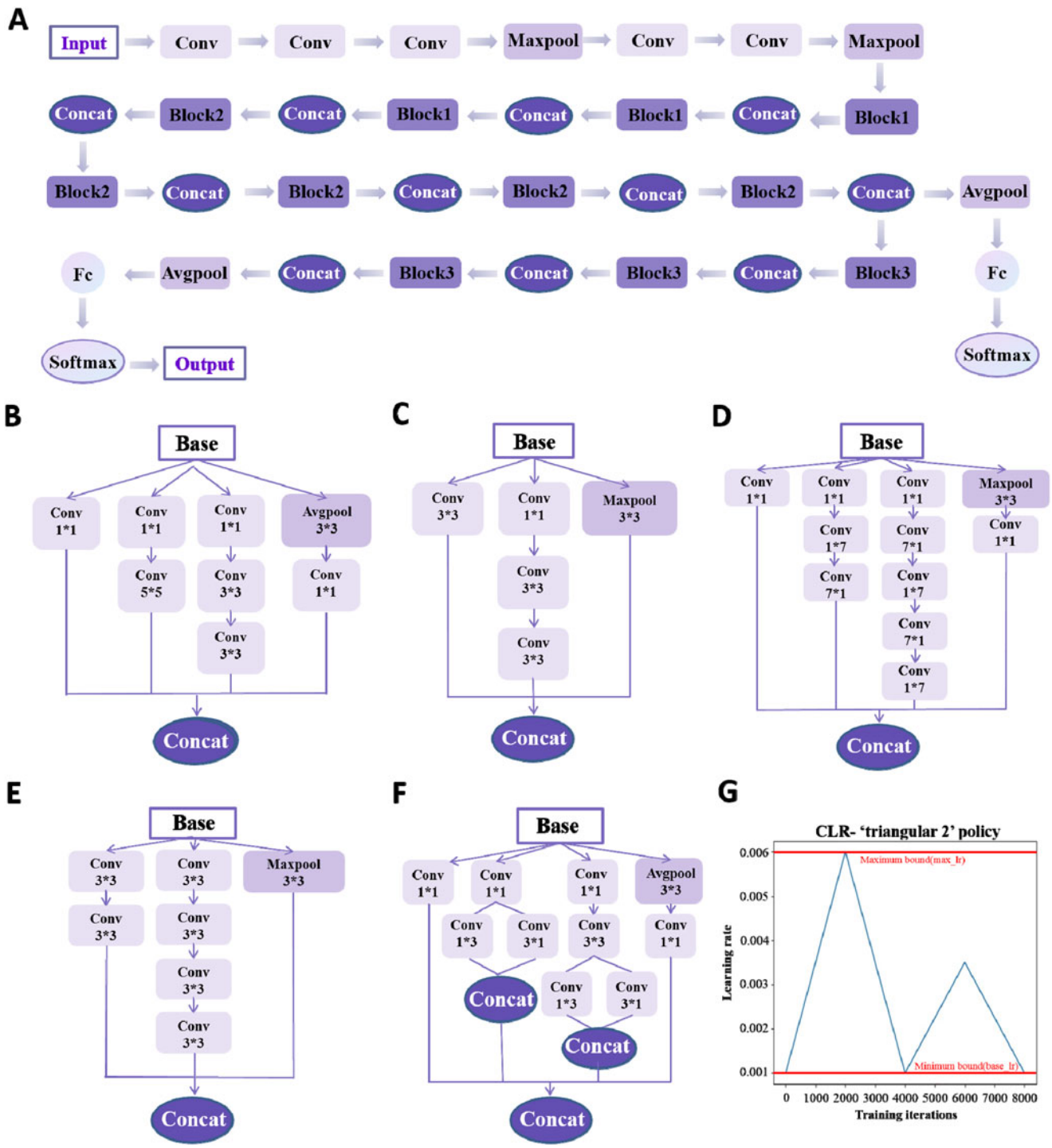


Fig. 3. Illustration of Inception-V3 model and triangular 2 learning rate policy. (**A**) The structural schematic view of Inception-V3 model. (**B**) The detail of modules in block 1. (**C**) and (**D**) are the details of modules in block 2. (**E**) and (**F**) are the details of modules in block 3. (**G**) Triangular 2 learning rate policy. The blue line represents learning rate values changing between red bounds. Max_LR means the maximum learning rate, is an upper bound. Base_LR means the minimum learning rate, is a lower bound.

3.2 Evaluation of PiTLiD Method

3.2.1 Dataset

To evaluate the method developed, we obtained apple disease data from the PlantVillage dataset [70]. The PlantVillage dataset is an open access repository of disease and healthy plant leaf images and corresponding labels. The dataset contains 12 healthy and 26 infected leaves of crop plants with a pure background. The images span 14 crop species. The dataset aims to help solve the problem of yield losses in crop plants due to infectious diseases through computer vision approaches [70]. The apple disease data in PlantVillage dataset includes four categories, i.e., black_rot, cedar_apple_rust, healthy, and apple_scab, as shown in Fig. 4**A**. They are abbreviated as black rot, rust, healthy, and scab respectively. To build benchmark data for evaluating the methods, 30 images from each category were chosen as training set and the remaining data were

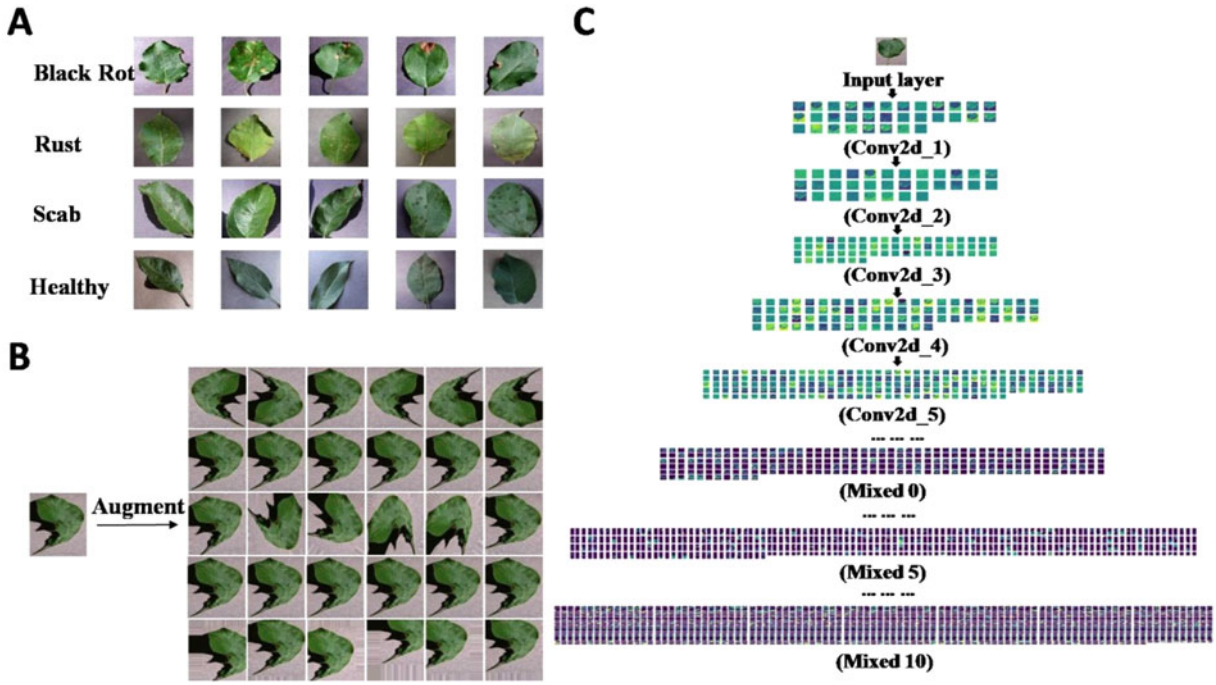


Fig. 4. (A) Examples of apple diseases leaf images for four classes. (**B**) Examples of data augmentation results of six different techniques. From left to right, each column from top to bottom is generated by flip, rescale, shear, shift, rotation, zoom, respectively. (**C**) Visualization of the convolutional operation. Conv2d_1 to Conv2d_5 are the outputs of the first five convolutional layers respectively. Mixed0, Mixed5, Mixed10 are the outputs of the first inception module, the fifth inception module, the last inception module respectively.

divided into validation set and testing set in a ratio close to 1:1. The data distribution is shown in Table 1.

3.2.2 Parameter Setting

The parameter setting with highest prediction accuracy was optimization algorithm 'RMSprop', batch size = 32 in training dataset and 16 in validation dataset. Each of the experimental runs for a total of 50 epochs, where the epoch is the number of the training iterations, steps = 200. CLR was used and the base LR is 0.001 and max LR is 0.006. During the training, the early stopping method was employed with patience = 14, which means the training will stop when 14 epochs the loss have no improvement. We also used l-2 regularization to avoid over-fitting. More parameter details can be seen in the code.

3.2.3 Illustration of Data Augmentation

The data augmentation has an essential impact on eliminating the occurrence of over-fitting, promoting the performance of model. In this experiment, we augmented training samples by using six data augmentation methods which are introduced in Part 2.2. Each batch of images input to the model will be randomly flipped, sheared, shifted, etc. The images are originally colored images of 256*256 sizes. For the Inception-V3 architecture, the images are resized to 299×299 pixels. Normalization of the input image is done by dividing all pixel values by 255 to make them compatible with the initial values of the network. As a result, the images obtained during each training epoch were not the same, which increases the training data in final. Hyperparameters of each augmentation skill are available in code. We took the leaf of black rot as an example and displayed its data augmentation results of five different techniques in Fig. 4**B**. In the figure, each column presents the data augmentation results of one technique.

TABLE 1
The Dataset for Training, Validation and Test

	Training	Validation	Test	Total
Black rot	30	296	295	621
Rust	30	123	122	275
Scab	30	300	300	630
Healthy	30	808	807	1645
Total	120	1527	1524	3171

3.2.4 Evaluation Criteria

In order to evaluate the model generation performance under apple disease datasets, we ran the model on validation data for 10 runs, the following five measures were calculated to evaluate the classification performance for the disease images, i.e., sensitivity (SEN), F1 score precision (PRC) and accuracy (ACC) [71], i.e.,

$$\begin{aligned} \mathrm{SEN} &= \mathrm{TP}/(\mathrm{TP}+\mathrm{FN}),\\ \mathrm{F1} &= 2*\mathrm{TP}/(2*\mathrm{TP}+\mathrm{FP}+\mathrm{FN}),\\ \mathrm{PRC} &= \mathrm{TP}/(\mathrm{TP}+\mathrm{FP}),\\ \mathrm{ACC} &= (\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}). \end{aligned}$$

In the above equations, TP, FP, TN and FN represent true positive false positive, true negative and false negative respectively.

3.2.5 Visualization of Feature Representation

To better understand convolutional mechanism, we output and visualized the result of the convolution in this model, as illustrated in Fig. 4**C**. Due to the page limit, the results only show a part. The input layer is an RGB image with a resolution of 256*256. As mentioned above, transfer learning is based on Inception-V3 model which contains convolutional layers and inception modules. In Fig. 4**C**, the layers from Conv2d_1 to Conv2d_5 correspond to the features extraction results of first five convolutional layers. In the first convolutional layer, 32 filters with size of 3*3 were used to extract features, thus 32 feature maps were produced and the size of each map is 127*127. Similarly, 32, 64, 80, 192 feature maps were produced in the subsequent four convolutional layers, with a size of 125*125, 125*125, 62*62, 60*60 respectively, as illustrated in Fig. 4**C**. The layer of Mix0 was obtained from the first inception module which contains 256 feature maps, with size of 29*29. Similarly, 768, 2048 feature maps were obtained from the fifth module and the last module, with a size of 14*14, 6*6 respectively. Finally, multi-scale feature representations were passed to the fully connected layers and flattened to 1-D vector, then 4 neurons output corresponds probability for four classes.

3.2.6. Different Fine-Tuning Settings Comparison

To accelerate learning, the pre-trained model with 51 layers was fine-tuned on PlantVillage dataset to identify and classify 4 categories of apple disease. There are three kinds of experiments to test three different fine-tuning settings on validation set. The model trained with the first setting is Sev_frozen in Fig. 5**A**, model trained with the second setting is Ten_frozen and model trained with the third setting is None_frozen. Fig. 5**A** shows that None_frozen which was fine-tuned all layers in model achieved the greatest prediction accuracy among all three models. The reason may be the generic feature of each disease image such as tints, shades, and textures are variant and extremely slight, None_frozen fine-tuned all layers which include the low layers that is beneficial to distinguish the slight difference. The fact ensures that fine-tuning all layers is consistently superior to freezing those layers for the pretrained model based on Inception-V3.

3.2.7. Comparison of Different Learning Rates

In this experiment, we compared three different learning rate methods on validation set to decide the best one. The accuracy scores of these experimental studies are shown in Fig. 5**B**. Con_LR represents the model which used the constant learning rate. The constant learning rate was maintained at 1e-3 throughout the model training process. Dec_LR represents the model which used the decay learning rate. The initial value of the decay learning rate is 1e-3, and every 20 epochs the decay learning rate decays at a rate of 1e-1 during training. Cyc_LR represents the model which used CLR. The base LR and max LR are 0.001 and 0.006 respectively, the CLR varies periodically between these two values during training, as illustrated in Fig. 3**G**.

As we can see from Fig. 5**B**, the accuracy of Con_LR model is worst, the accuracy of the Dec_LR model rose steadily, the maximum accuracy of 97.3% was achieved when the epoch reached 47. Only CLR was used in Cyc_LR model can the accuracy be maximized, CLR could break through the saddle point to achieve higher accuracy of 99.31% than the decay learning rate in 30 and 49 epochs. Thus, CLR is the best one for diseases classification.

3.2.8. Classification Performance

After the hyper-parameters and fine-tuning setting were fixed, the final model was tested over 10 runs on the test set with the same parameter setting to prove stability and the result is illustrated in Fig. 5**C**. We measured different classification performance metrics such as accuracy, sensitivity, precision and the F1 score of each run. As shown in Fig. 5**C**, PiTLiD yielded an accuracy of 99.45 $\pm$ 0.17%, a sensitivity of 99.10 $\pm$ 0.23%, a precision of 98.84 $\pm$ 0.31%, and an F1 score of 99.00%$\pm$ 0.23%. The fluctuation of each measure is small. So, the method established in this paper has high and stable classification performance.

3.2.9. Comparison With Current Existing Approaches

An analysis is conducted to check the efficacy of PiTLiD in performing the plant leaf disease classification with small sample size by comparing it with other models. To perform this task, the classification performance of PiTLiD is evaluated against a base model called LeNet and three models with different training approaches as follows [72]. The LeNet architecture is a simple CNN model which has been considered as a baseline for a variety of tasks [72], [73]. Training from scratch is the first training method that all layers in model are initialized with random weights and parameters. The second training method, deep feature extraction, obtained the output of the feature extraction module of pretrained model, then the output is delivered to a classifier to complete classification. Retraining is the third training method that last layers in model were retrained with random weights and first seven layers were retrained with pretrained weight of corresponding layers. Accuracy, sensitivity, precision and the F1 score were calculated for performance measurement.

We presented the comparison results of our approach with several models in terms of classification performance in Fig. 5**D**. We can observe that the best performance is obtained by using PiTLiD, which has more improvement compared to the next best approach. There are several reasons for the good performance: PiTLiD used the parameters of the pretrained model which has already learned features from a massive dataset as initialization, so, the process of training network will be fast because it doesn't have to start from scratch; the augmented training dataset was utilized to fine-tune whole model to learn properties of new images of our apple disease dataset so model can become more appropriate to the apple disease classification task.

3.2.10. Predictive Capability on Other Species Data

To further analyze the prediction performance of PiTLiD on other small species diseases data, we conducted experiments to test PiTLiD on other datasets. Two diseases datasets of grape and peach were fed into PiTLiD respectively for classification. The number of training samples of both

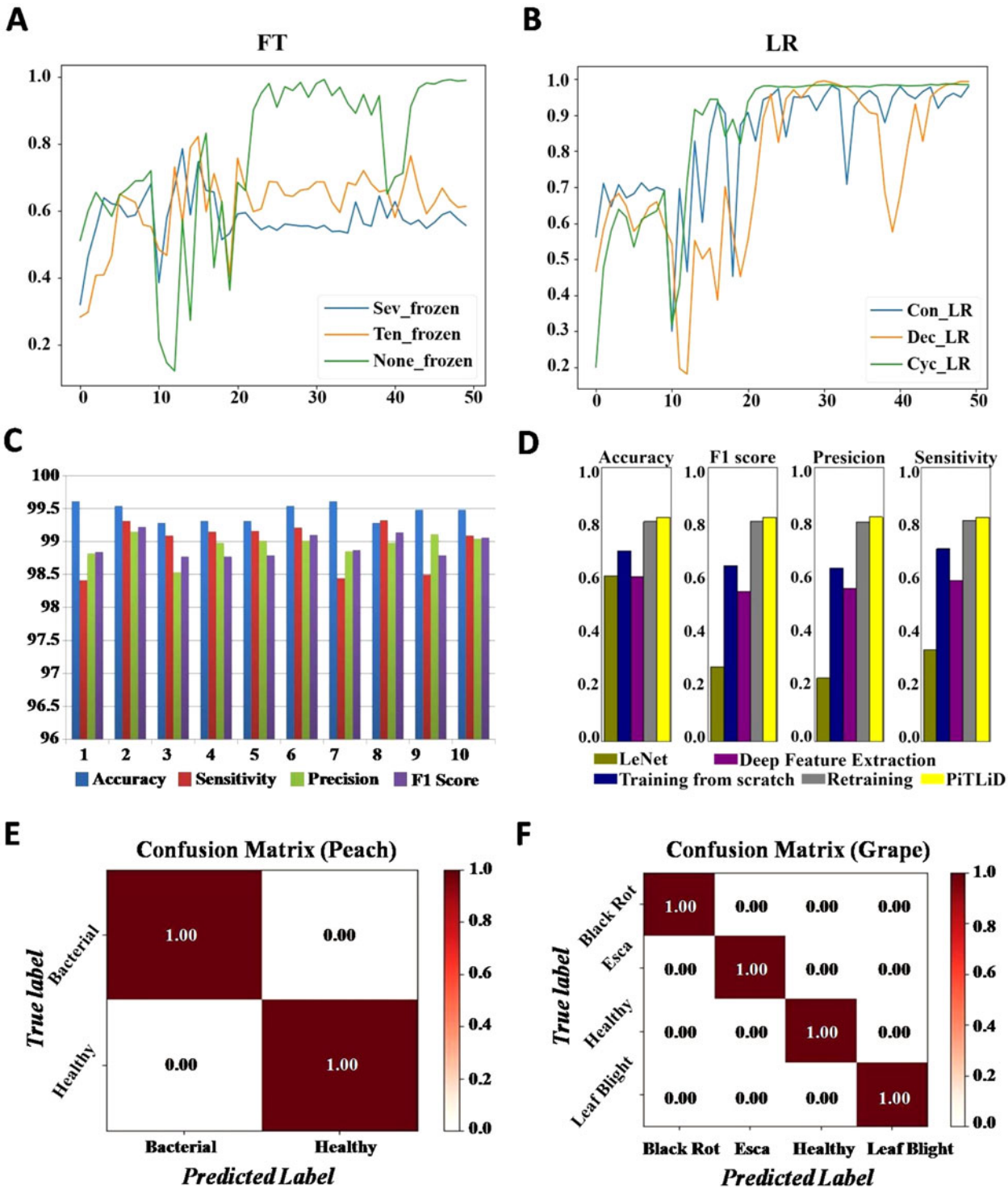


Fig. 5. Results of the evaluations. (**A**) Comparison of different fine-tune settings. The abscissa is the number of epochs, and the ordinate is the value of accuracy. (**B**) Comparison of different learning rates. The abscissa is the number of epochs, and the ordinate is the value of accuracy. (**C**) Ten runs of PiTLiD trained on the test dataset. ACC, SEN, PRC and F1 mean accuracy, sensitivity, precision and F1 score respectively. (**D**) The bar graph for the comparison of five approaches. (**E-F**) The confusion matrices of model trained on insufficient diseases datasets of grape and peach. Rows correspond to truth labels, while columns correspond to predicted labels. Cells are colored to indicate cell values according to the legend at the right.

datasets is 30. Images from the two datasets were pre-processed like the apple diseases datasets (see METHODS). The results were displayed in the form of confusion matrix, which can see in Figs. 5**E** and 5**F**. There are high degrees of confusion between corresponding categories. Ideal performance is displayed by the values of 1.00 along with the matrix diagonal and the values of 0.00 everywhere else. This result illustrates that PiTLiD has good performance when applied to other datasets.

4 Conclusion

In this work, we proposed an Inception-V3-based transfer learning method called PiTLiD and applied it to the apple

diseases classification task. PiTLiD is aim to address the small sample size problem of the training data, without the demand of a large number of phenotype samples that traditional CNN requires. The results of PiTLiD with pathological images show a promising performance with an accuracy of 99.45 $\pm$ 0.17%. Moreover, the classification result of PiTLiD applied to other small samples verified its robustness. Future work may include the following three points. (1) Combining with other deep learning models, such as ResNet, DenseNet, SqueezeNet and so on to obtain extra performance. (2) Improving data augmentation techniques. Current data augmentation technologies largely depend on existing disease data and so how to use more advanced techniques to directly generate new data after learning disease images features. (3) Applying this approach to other fields to solve the problem that only a small number of training samples with accurate labels can be obtained.

REFERENCES

[1] J. Fan, Y. Zhang, W. Wen, S. Gu, X. Lu, and X. Guo, "The future of Internet of Things in agriculture: Plant high-throughput phenotypic platform," *J. Cleaner Prod.*, vol. 280, 2021, Art. no. 123651.

[2] S. Kolhar and J. Jagtap, "Plant trait estimation and classification studies in plant phenotyping using machine vision – a review," *Inf. Process. Agriculture*, 2021.

[3] S. Zhou, H. Mou, J. Zhou, J. Zhou, H. Ye, and H. T. Nguyen, "Development of an automated plant phenotyping system for evaluation of salt tolerance in soybean," *Comput. Electron. Agriculture*, vol. 182, 2021, Art. no. 106001.

[4] J. Zhang, Y. Rao, C. Man, Z. Jiang, and S. Li, "Identification of cucumber leaf diseases using deep learning and small sample size for agricultural Internet of Things," *Int. J. Distrib. Sensor Netw.*, vol. 17, no. 4, 2021, Art. no. 15501477211007407.

[5] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Comput. Electron. Agriculture*, vol. 161, pp. 280–290, 2019.

[6] P. Tm, A. Pranathi, K. SaiAshritha, N. B. Chittaragi, and S. G. Koolagudi, "Tomato leaf disease detection using convolutional neural networks," in *Proc. 11th Int. Conf. Contemporary Comput.*, 2018, pp. 1–5.

[7] V. A. Natarajan, M. M. Babitha, and M. S. Kumar, "Detection of disease in tomato plant using deep learning techniques," *Int. J. Modern Agriculture*, vol. 9, no. 4, pp. 525–540, 2020.

[8] Z. Li, R. Guo, M. Li, Y. Chen, and G. Li, "A review of computer vision technologies for plant phenotyping," *Comput. Electron. Agriculture*, vol. 176, 2020, Art. no. 105672.

[9] V. Sravan, K. Swaraj, K. Meenakshi, and P. Kora, "A deep learning based crop disease classification using transfer learning," in *Materials Today: Proceedings*, Amsterdam, Netherlands: Elsevier, 2021.

[10] D. Chen et al., "Predicting plant biomass accumulation from image-derived parameters," *GigaScience*, vol. 7, no. 2, pp. 1–13, 2018.

[11] J. Casadesús and D. Villegas, "Conventional digital cameras as a tool for assessing leaf area index and biomass for cereal breeding: Conventional digital cameras for cereal breeding," *J. Integrative Plant Biol.*, vol. 56, no. 1, pp. 7–14, 2014.

[12] S. Jin et al., "Deep learning: Individual maize segmentation from terrestrial lidar data using faster r-cnn and regional growth algorithms," *Front. Plant Sci.*, vol. 9, 2018, Art. no. 866.

[13] J. R. Ubbens and I. Stavness, "Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks," *Front. Plant Sci.*, vol. 8, 2017, Art. no. 1190.

[14] Y. Weng, R. Zeng, C. Wu, M. Wang, X. Wang, and Y. Liu, "A survey on deep-learning-based plant phenotype research in agriculture," *Scientia Sinica Vitae*, vol. 49, no. 6, pp. 698–716, 2019.

[15] S. Taghavi Namin, M. Esmaeilzadeh, M. Najafi, T. B. Brown, and J. O. Borevitz, "Deep phenotyping: Deep learning for temporal phenotype/genotype classification," *Plant Methods*, vol. 14, no. 1, 2018, Art. no. 66.

[16] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[17] N. Coudray et al., "Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning," *Nature Med.*, vol. 24, no. 10, pp. 1559–1567, 2018.

[18] Y. Han, S. Ma, Y. Xu, L. He, S. Li, and M. Zhu, "Effective complex airport object detection in remote sensing images based on improved End-to-End convolutional neural network," *IEEE Access*, vol. 8, pp. 172652–172663, 2020.

[19] Y. Liu and M. Lapata, "Learning structured text representations," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 63–75, 2018.

[20] Z. You, T. Yang, and M. Jin, "Multi-channel deep 3D face recognition," 2020, *arXiv:2009.14743*.

[21] A. Khanna and S. Kaur, "Evolution of Internet of Things (IoT) and its significant impact in the field of precision agriculture," *Comput. Electron. Agriculture*, vol. 157, pp. 218–231, 2019.

[22] M. Brahimi, M. Arsenovic, S. Laraba, S. Sladojevic, K. Boukhalfa, and A. Moussaoui, "Deep learning for plant diseases: Detection and saliency map visualisation," in *Human Machine Learning*, J. Zhou and F. Chen, eds., Cham, Switzerland: Springer, 2018, pp. 93–117.

[23] D. P. Hughes, "Deep learning for image-based cassava disease detection," *Front. Plant Sci.*, vol. 8, 2017, Art. no. 7.

[24] A. K. Rangarajan, R. Purushothaman, and A. Ramesh, "Tomato crop disease classification using pre-trained deep learning algorithm," *Procedia Comput. Sci.*, vol. 133, pp. 1040–1047, 2018.

[25] J. Ma, K. Du, F. Zheng, L. Zhang, Z. Gong, and Z. Sun, "A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network," *Comput. Electron. Agriculture*, vol. 154, pp. 18–24, 2018.

[26] G. Zhang, T. Xu, Y. Tian, H. Xu, J. Song, and Y. Lan, "Assessment of rice leaf blast severity using hyperspectral imaging during late vegetative growth," *Australas. Plant Pathol.*, vol. 49, no. 5, pp. 571–578, 2020.

[27] S. V. Mirnezami et al., "Automated trichome counting in soybean using advanced image-processing techniques," *Appl. Plant Sci.*, vol. 8, no. 7, 2020, Art. no. e11375.

[28] A. dos Santos Ferreira, D. Matte Freitas, G. Gonçalves da Silva, H. Pistori, and M. Theophilo Folhes, "Weed detection in soybean crops using convnets," *Comput. Electron. Agriculture*, vol. 143, pp. 314–324, 2017.

[29] C. S. Bekkering, J. Huang, and L. Tian, "Image-based, organ-level plant phenotyping for wheat improvement," *Agronomy*, vol. 10, no. 9, 2020, Art. no. 1287.

[30] W. V. Marset, D. S. a. P'erez, C. A. D'iaz, and F. Bromberg, "Deep learning for 2D grapevine bud detection," 2020, *arXiv:008.11872*.

[31] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers," *Plants*, vol. 9, no. 10, 2020, Art. no. 1319.

[32] N. D'Souza R, P. Y. Huang, and F. C. Yeh, "Structural analysis and optimization of convolutional neural networks with a small sample size," *Sci. Rep.*, vol. 10, no. 1, Jan. 2020, Art. no. 834.

[33] R. Keshari, M. Vatsa, R. Singh, and A. Noore, "Learning structure and strength of cnn filters for small sample size training," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9349–9358.

[34] W. Zhao, "Research on the deep learning of the small sample data based on transfer learning," in *Proc. AIP Conf.*, 2017, Art. no. 020018.

[35] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agriculture*, vol. 161, pp. 272–279, 2019.

[36] D. Mishkin and J. Matas, "All you need is a good init," 2016, *arXiv:1511.06422*.

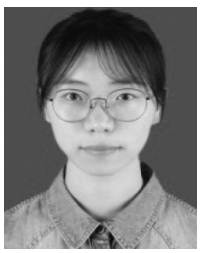
[37] M. Torres Torres, M. Valstar, C. Henry, C. Ward, and D. Sharkey, "Postnatal gestational age estimation of newborns using small sample deep learning," *Image Vis. Comput.*, vol. 83-84, pp. 87–99, 2019.

[38] A. J. R. Joe and N R., "Scaling transform methods for compressing a 2D graphical image," *Adv. Comput.: An Int. J.*, vol. 4, no. 2, pp. 41–50, 2013.

[39] T. Zhang, J. Liang, Y. Yang, G. Cui, L. Kong, and X. Yang, "Antenna deployment method for multistatic radar under the situation of multiple regions for interference," *Signal Process.*, vol. 143, pp. 292–297, 2018.

[40] Y.-D. Zhang et al., "Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation," *Multimedia Tools Appl.*, vol. 78, no. 3, pp. 3613–3632, 2019.

[41] E. Okafor, R. Smit, L. Schomaker, and M. Wiering, "Operational data augmentation in classifying single aerial images of animals," in *Proc. IEEE Int. Conf. INnovations Intell. Syst. Appl.*, 2017, pp. 354–360.
[42] Y. Yang et al., "Deployment of multistatic radar system using multi-objective particle swarm optimisation," *IET Radar Sonar Navigation*, vol. 12, no. 5, pp. 485–493, 2018.
[43] J. Wu, W. Pu, Y. Huang, J. Yang, and H. Yang, "Bistatic forward-looking SAR focusing using ω–k based on spectrum modeling and optimization," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 11, pp. 4500–4512, Nov. 2018.
[44] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2530–2538.
[45] D. Xiao, Y. Huang, C. Qin, Z. Liu, Y. Li, and C. Liu, "Transfer learning with convolutional neural networks for small sample size problem in machinery fault diagnosis," *Proc. Inst. Mech. Engineers Part C: J. Mech. Eng. Sci.*, vol. 233, no. 14, pp. 5131–5143, 2019.
[46] E. Li, J. Xia, P. Du, C. Lin, and A. Samat, "Integrating multilayer features of convolutional neural networks for remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 10, pp. 5653–5665, Oct. 2017.
[47] S. Chauhan, L. Vig, M. De Filippo De Grazia, M. Corbetta, S. Ahmad, and M. Zorzi, "A comparison of shallow and deep learning methods for predicting cognitive performance of stroke patients from MRI lesion images," *Front. Neuroinform.*, vol. 13, 2019, Art. no. 53.
[48] J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: A review," *Plant Methods*, vol. 17, no. 1, Feb. 2021, Art. no. 22.
[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[50] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, 2018.
[51] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*, F. F. Soulié and J. Hérault, eds., Berlin, Germany: Springer, 1990, pp. 227–236.
[52] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[53] M. Lin, Q. Chen, and S. Yan, "Network in network," 2014, *arXiv:1312.4400*.
[54] Z. Xiao et al., "In silico design of MHC class i high binding affinity peptides through motifs activation map," *BMC Bioinf.*, vol. 19, 2018, Art. no. 516.
[55] M. Hussain, J. J. Bird, and D. R. Faria, "A study on cnn transfer learning for image classification," in *Proc. Adv. Comput. Intell. Syst.*, 2019, pp. 191–202.
[56] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[57] A. Kaya, A. S. Keceli, C. Catal, H. Y. Yalic, H. Temucin, and B. Tekinerdogan, "Analysis of transfer learning for deep neural network based plant classification models," *Comput. Electron. Agriculture*, vol. 158, pp. 20–29, 2019.
[58] C. Wang et al., "Pulmonary image classification based on inception-v3 transfer learning model," *IEEE Access*, vol. 7, pp. 146533–146541, 2019.
[59] J. Deng, W. Dong, R. Socher, L. Li, L. Kai, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[60] A. van Opbroek, M. A. Ikram, M. W. Vernooij, and M. de Bruijne, "Transfer learning improves supervised image segmentation across imaging protocols," *IEEE Trans. Med. Imag.*, vol. 34, no. 5, pp. 1018–1030, May 2015.
[61] N. Tajbakhsh et al., "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1299–1312, May 2016.
[62] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*.
[65] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, eds., Berlin, Germany: Springer, 2012, pp. 437–478.
[66] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," 2018, *arXiv:1803.09820*.
[67] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?," 2019, *arXiv:1908.01878*.
[68] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2017, pp. 464–472.
[69] J. Li and X. Yang, "A cyclical learning rate method in deep learning training," in *Proc. Int. Conf. Comput. Inf. Telecommunication Syst.*, 2020, pp. 1–5.
[70] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing," 2015, *arXiv:1511.08060*.
[71] A. A. AlBeladi and A. H. Muqaibel, "Evaluating compressive sensing algorithms in through-the-wall radar via F1-score," *Int. J. Signal Imag. Syst Eng.*, vol. 11, no. 71, 2018, Art. no. 8.
[72] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[73] H. A. Atabay, "Binary shape classification using convolutional neural networks," *IIOAB J.*, vol. 7, no. 5, pp. 332–336, 2016.



Kangchen Liu is currently working toward the PhD degree in plant bioinformatics with the University of Chinese Academy of Sciences, China and Wuhan Botanical Garden of Chinese Academy of Sciences, China. Her research interests include bioinformatics, plant phenomics, and machine learning.

Xiujun Zhang received the PhD degree in bioinformatics and systems biology from Shanghai University, China and jointly from the University of Angers, France in 2013. He has ever worked as a postdoc with the University of California at Los Angels, USA and Nanyang Techology Universtiy, Singapore from 2013 to 2016. He is currently a principle investigate researcher with the Wuhan Botanical Garden, Chinese Academy of Sciences, China. His research interests include bioinformatics, systems biology, and computational biology.