Method Article

Research and development of invasive plant recognition system based on deep learning

ABSTRACT

A major ecological issue that has seriously harmed both human society and the environment is the invasion of alien plants. To stop the invasion of alien plants, it is crucial to create an effective and precise monitoring and early warning system. The mobile client, server, and deep learning-based invasive plant recognition model for APP are all included in this study. The identification result can be acquired in 1 to 2 seconds after downloading the plant image from the APP, uploading it to the server, and using the convolutional neural network (CNN). The system had an average accuracy of 90.39% on the test set thanks to data augmentation and enhanced networks. The deep learning-based invasive plant identification system created in this study has demonstrated through experiments that it may effectively support botanical research and ecological environment monitoring.

Keywords: invasive plant, deep learning, mobile client, CNN, image recognition

1. INTRODUCTION

The ecological problem of alien plant invasion has seriously harmed both human society and the natural world. Exotic plant invasion has been linked to numerous studies that demonstrate how it negatively affects the stability, ecosystem function, and species richness of nearby ecosystems. Moreover, the invasion of alien plants will impact the management of water resources, urban forestry, and agricultural productivity [1]. In order to safeguard the ecological environment and promote sustainable development, it is crucial to effectively prevent and regulate the invasion of alien plants.

Conventional techniques for identifying alien plants mostly 5rely on personal professional experience, reference materials, web images for comparison, or obtaining professional identification. Yet, because there are so many different exotic plants, there is a lot of variance among the same species of plants despite similarities in appearance. As a result, this artificial identification method for exotic plants is ineffective and sensitive to human bias.

In the past 20 years, a large number of academics both domestically and internationally have conducted research on the automatic recognition of alien plants based on photographs thanks to the development and use of image processing technology. Initially, high-quality photos of exotic plants are acquired using a variety of image acquisition tools. Next, image features are extracted, namely SIFT [2] and HOG [3] in the case of local features and color [4], shape [5], or texture [6] for global features. On a small number of types and sample sets, the traditional pattern recognition techniques mentioned above can produce good recognition results. Several study findings, however, have not been used in actual production because of the model's limited resilience and weak generalization ability, which result from the requirement to manually create features.

Deep learning techniques have gained increasing attention in recent years for their use in tracking alien plants. The accuracy and effectiveness of identifying exotic plants is substantially improved by deep learning techniques like convolutional neural networks (CNN) and recurrent neural networks (RNN), which automatically extract information from collected photos [7]. When compared to conventional image processing-based techniques, CNNs enable effective analysis of image texture, or the contextual data of numerous nearby pixels. The effective analysis of these textures is made possible by CNNs' self-learning abilities, which can also reveal the vital leaf and canopy characteristics needed to recognize vegetation communities or species [8]. Hence, CNN will fundamentally alter our ability to categorize and identify plants as a result of the development of high-resolution sensor technologies.

CNN were created with the purpose of classifying images. Software like Flora Incognita or Plannet, which can name a species to plant images, are examples of plant recognition. Such a Network automatically pulls contextual elements from a dataset of photos, learns which of these features (for example, leaf shapes or attributes of the flowers), and then categorizes each image based on observations. The self-learning capability of CNNs is a huge advantage in terms of processing power and automation, as no feature engineering procedure is necessary, given the variety of approaches and scales for representing backgrounds. CNN were created with the purpose of classifying images. Software like Flora Incognita or Plannet, which can name a species to plant images, are examples of plant recognition. Such a Network automatically pulls contextual elements from a dataset of photos, learns which of these features (for example, leaf shapes or attributes of the fowers), and then categorizes each image based on observations. The self-learning capability of CNNs is a huge advantage in terms of processing power and automation, as no feature engineering procedure is necessary, given the variety of approaches and scales for representing backgrounds. Several successive pooling procedures that combine feature maps from convolutions to coarser spatial scales, hence boosting the robustness and effectiveness of the network, are a crucial part of a CNN architecture to recognize these features. Only (aggregated) information, i.e., if characteristics indicating the target class are evident everywhere in the image, will be present in the network's final layer. According to CNN, there are few studies on the monitoring of plant invasion in the pertinent literature [9]; instead, studies on plant identification and classification, agricultural pest identification, and plant leaf disease research are more prevalent.

In order to address the issue that farmers or volunteer plant protection workers cannot identify invasive plants in the field, this paper suggests monitoring plant invasion based on CNN and uses deep learning and Android technology to develop a terminal-oriented intelligent identification and monitoring system for plant invasion. In particular, taking images of invasive plants and uploading them to the system's Android mobile terminal can enable automatic identification and give consumers individualized advice on how to control those plants.

2. SYSTEM DESIGN

Figure 1 depicts the general layout of the mobile terminal-oriented alien plant invasion monitoring system. An agricultural pest identification model, a cloud server, and a mobile APP make up the system. Images of plants are captured by the mobile APP, which then transmits them over the network to a cloud server. The invasive plants are identified using the identification model placed on the cloud server, and the mobile end receives the results.

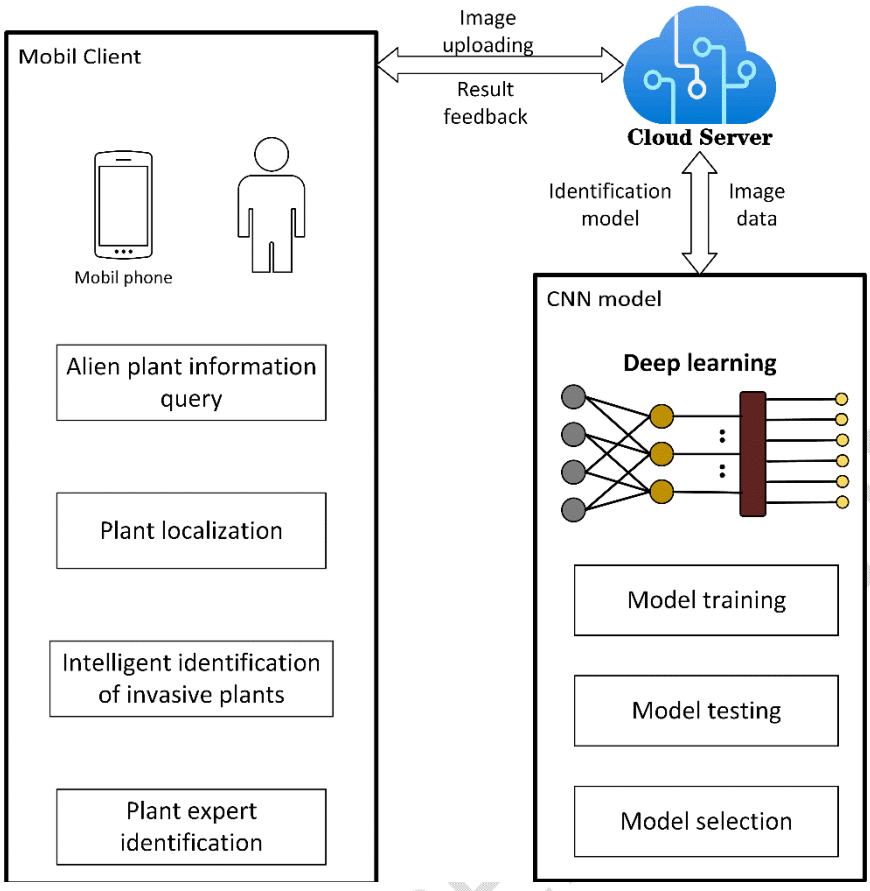

Fig. 1. Design of alien plant invasion monitoring system for mobile terminal

2.1 Invasive plant species and data sets

This study's datasets were primarily gathered via the Internet and field photography, which involved simultaneously photographing a variety of plants while collecting open-source image databases and internet-related photographs. According to plant species, all photos were categorized, sorted, and manually marked. In Figure 2, a few pictures of invasive plants are displayed. There are a total of 25 categories in this collection, which includes photos of 9 different common plants and 16 different exotic plant kinds. There are 12606 RGB plant picture samples altogether after some unbalanced samples have been extended and enhanced. Also, the collection of invasive plant picture data is divided at random into the training set and the test set, which correspond to an 8:2 ratio.

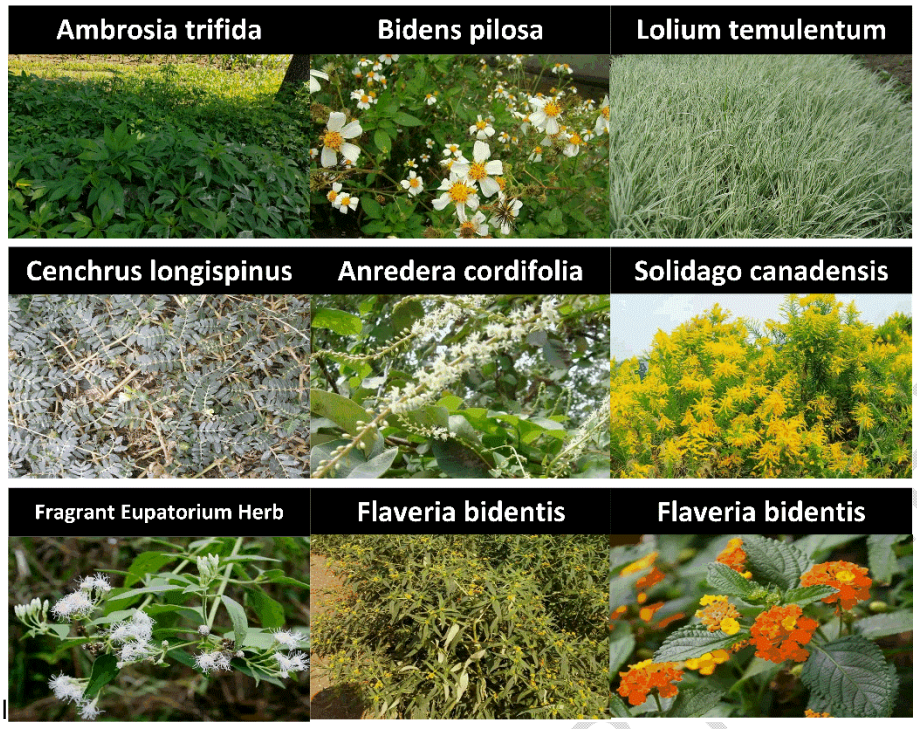

Fig. 2. Examples of alien invasive plant images

Use the python program to encode, and view the number of each category in the gathered plant and invasive plant picture data set. Manually name the 25 species in the data set. The breakdown of the training set's sample count for each category is shown in Figure 3.

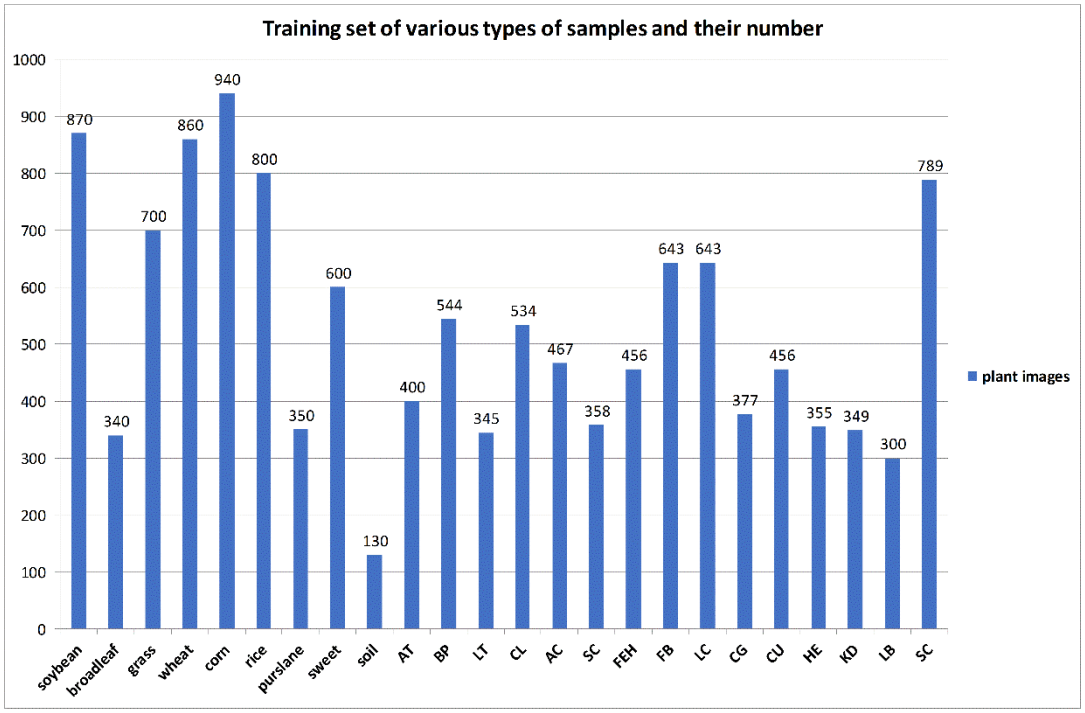

Fig. 3. Quantity details of samples in each category in training set

2.2 image pre-processing

It is crucial to create an accurate and comprehensive data set because deep learning algorithms often need a lot of data to work well. To train the model for precise monitoring and prediction in the alien plant invasion monitoring system, the dataset should contain photos of various alien plant species in various regions. Nevertheless, the source photos frequently have issues like noise, occlusion, and uneven lighting, which will negatively impact the model's performance during training. In order to enhance image quality and model accuracy before deep learning, the dataset's images must be pre-processed.

The image is 256 x 256 pixels in its original size. The scale of the image is evenly changed to 227 X 227 pixels in order to match the input size of the network. A Python script is then used to perform random flipping, random angle rotation, and other data improvement actions on selected imbalanced samples. Also, we must normalize the gamut of pixel intensity values found in the dataset's leaf image before training the CNN network. This step is required because the recovered feature vector from the input image should have all of its dimensions in the same intensity range. This enables the training phase of our CNN model to converge more quickly.
The method of image normalization is to subtract the value $I(i,j)$ of each pixel from the average value μ of the input image, and then divide it by the standard deviation σ of the input image. The distribution of the output pixel intensity value will be similar to a zero-centered Gaussian curve. We use the following formula (1) to normalize each image in the training set:

$$O(i,j) = \frac{I(i,j) - \mu}{\sigma} \quad (1)$$

Information Among them, I and O are the input image and the output image respectively, i, j are the pixel index to be normalized

2.3 Plant classification method based on improved DenseNet model

2.3.1 DenseNet Model

Huang et al. presented the DenseNet convolutional neural network model in 2017 [10]. The network makes use of the ResNet structure to connect the front and back layers, which facilitates gradient back propagation and the training of deeper networks. The DenseNet model can achieve feature reuse, minimize the gradient disappearance issue, simplify the model by using fewer parameters, and increase the propagation and use rates of features. DenseNet is made up of transition layers and a number of dense blocks. Every layer in a dense block takes input from all layers before it, and each layer then generates an output that is connected to all inputs before it and passed to the following layer. The effectiveness of feature propagation is increased by this thick connectivity, which enables each layer to make use of the knowledge from all preceding levels. These two components are then introduced in turn.

More radical than ResNet's connection scheme is DenseBlock, a densely connected block in the DenseNet convolutional neural network architecture. The residual network serves as the foundation for DenseBlock's evolution. Each layer of the residual network is connected to the

preceding one or two layers using this connection approach. By adding, or increasing the input from the preceding layer, the short-circuit connection is created:

$$x_l = H_l(x_{l-1}) + x_{l-1} \quad (2)$$

In order to achieve feature reuse, DenseNet uses a dense connection technique to connect all layers to one another, with each layer being connected to all preceding layers in the channel dimension:

$$x_l = H_l(x_0, x_1, \ldots, x_{l-1}) \quad (3)$$

x_l is the input of the L layer, which takes the convolution feature $x_0 \sim x_{l-1}$ from all the preceding layers as input; $(x_0, x_1, \ldots, x_{l-1})$ is the splicing operation of the output feature map from the input layer to the L-1 layer; and $H_l(*)$ is a nonlinear transformation function, which combines the functions of the BN layer, ReLU layer, and convolution layer. Figure 4 depicts the architecture of the dense connection module and the common convolution module.

In the DenseNet model, the Transition module's function is to compress the model. 1 X 1 convolutional layer is added between the 2 X 2 pooling layers between the dense blocks of two identical feature maps to minimize the size. The number of channels transferred to the next dense block is cut in half, and there are fewer parameters, which helps speed up the network model's convergence when the compression factor in the DenseNet network is set to 0.5.

2.3.2 DenseNet-SPP Model

He et al. (2015) presented a pyramid pooling SPP (Spatial Pyramid Pooling) [11] structure to address the issue of input image size limitation in image classification applications. Before to the completely linked layer, a structure known as the SPP structure is used to create the network at various scales. To improve the network's ability to recognize objects of various scales, the data are pooled and features from multiple scales are combined. Since there are many different scales and shapes of invasive plants, we believe that adding the SPP structure to the DenseNet network can further enhance the network's performance.

A multi-layer feature map's channel is adjusted through the convolution of the last layer, and SPP structure is added before the full connection layer and the final convolution layer. Each feature map in this study is separated into several sizes based on 16 X 16, 4 X 4, and 1 X 1. The feature map is converted into a fixed length feature vector through the splicing channel and utilized as the input of the entire connection layer after the grid and Max Pooling operation on each grid. Each feature map is coupled to a 1 X 1 convolution layer to compress and alter the channel dimension. Often, the SPP structure pools via different size windows to obtain feature maps with varied resolutions. In Figure 5, the SPP structure is displayed.

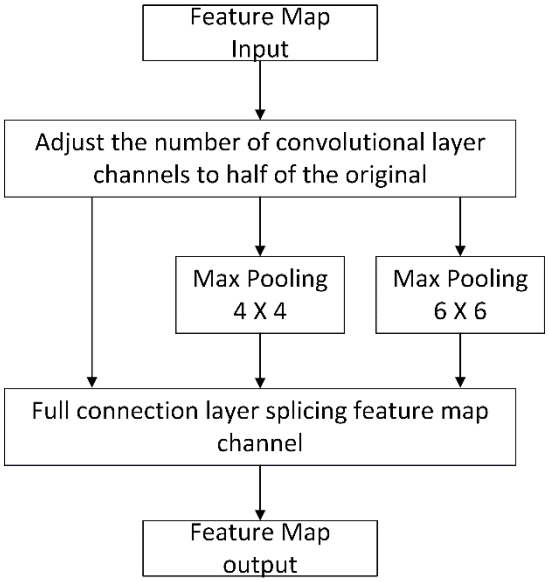

Fig. 5. SPP structure diagram illustration

An SPP layer is added to the last dense block of DenseNet to pool multi-scale features. The feature map is then input into the fully connected layer for classification jobs, although the fully connected layer needs to be changed based on the data set. the connecting layer's output dimension and category count. The 25 categories used to categorize the invasive plants under study in this paper result in 25 outputs for the completely connected layer. This research specifies the maximum pooling scales of 6 X 6 and 4 X 4 respectively to adapt to the various sizes of invasive plants while taking into account the proportion of invasive plants on the image. In Figure 6, DenseNet's entire network architecture with the SPP structure is displayed.

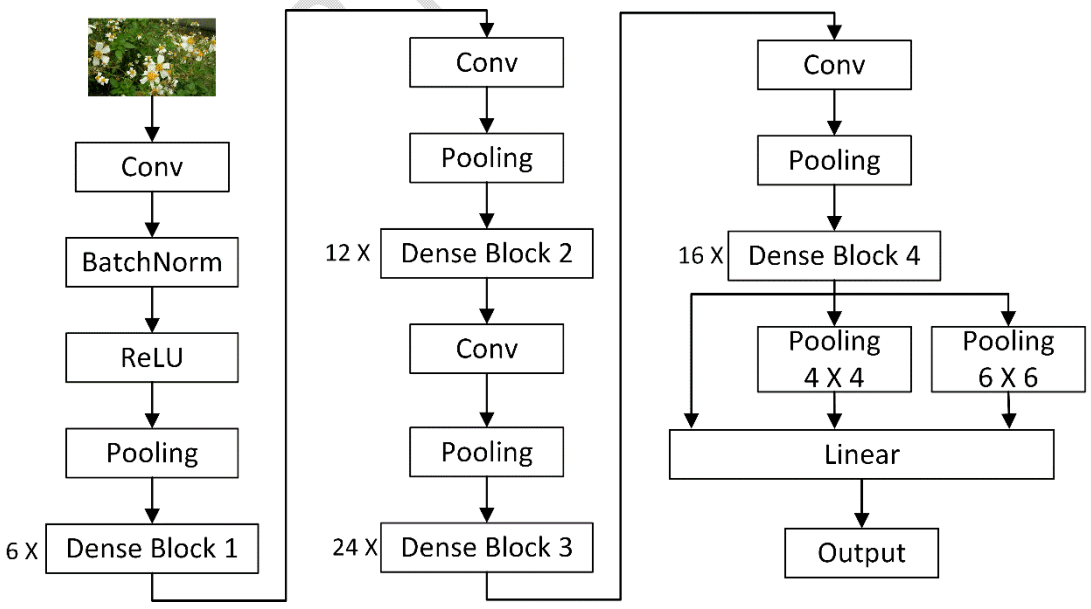

Fig. 6. DenseNet network structure diagram with SPP

3. IMPLEMENTATION

The details of the intrusion plant detector's deployment on a cloud server and a mobile terminal are introduced in this part.

3.1 DenseNet-SPP model Implementation

This paper investigates and enhances the image classification network based on the previous research. The network structure $f(\theta)$, loss function L, and learning rate L_R are trained using the invasive plant picture data set $X(x_1, x_2, \dots, x_n)$ and the matching m category $Y(y_1, y_2, \dots, y_m)$. The following describes the algorithm training method.

i) Based on the pre-defined Batch size for normalizing pre-processing, choose training batch $X(x_1, x_2, \dots, x_n)$ from the training set. According to the following Formula (4) computation, the RGB three channels for each image of an invasive plant in the training batch are assigned to the mean value $\mu_{r,g,b}$ = (0.485,0.456,0.406) and the standard deviation $\sigma_{r,g,b}$ = (0.229,0.226,0.225) , respectively.

$$O_r = \frac{I_r - \mu_r}{\sigma_r};\ O_g = \frac{I_g - \mu_g}{\sigma_g};\ O_b = \frac{I_b - \mu_b}{\sigma_b} \tag{4}$$

In formula (4), I_r、I_g、I_b are the RGB three-channel pixel value of the image before normalization, and O_r、O_g、O_b are the RGB three-channel pixel value of the normalized image

ii) The network receives the normalized training batch $X_1'(x_a', \dots, x_b')$ and begins forward propagation to extract features. The feature map acquired by the nth layer in DenseBlock is created by connecting and convolving the feature maps in all levels up to n-1. The computation's formula can be seen in (5)

$$M_n = F(M_1 \oplus M_2 \oplus \cdots \oplus M_{n-1}) \tag{5}$$

In Equation (5), M_n represents the output feature of the nth layer of the network, F represents the convolution operation, and $\oplus$ represents the interconnection operation between the features, so it can be obtained that the n-th layer network has $(n^2 + n)/2$ connection operations

iii) Following the network's training of the training batch, the corresponding category Y_1' is determined. The loss L_1 is then determined in accordance with the predicted category Y_1' and the actual category Y_1, as indicated in formula (6):

$$\begin{aligned} \mathrm{Y}_1^{'} &= f(\theta, X_1^{'}) \\ L_1 &= L(Y_1, \mathrm{Y}_1^{'}) \end{aligned} \tag{6}$$

In this paper, the cross-entropy loss function is used in the training process to solve the problem of unbalanced positive and negative samples. The cross-entropy loss function is shown in formula (7):

$$L_1 = -\frac{1}{\text{b-a}} \sum_{i=1}^{b-a} \sum_{c=1}^{m} p_{ic} \log(q_{ic}) \quad (7)$$

In Equation (7), b-a represents the number of samples in each training batch, c represents the number of categories in total m, and p_{ic} is a symbolic function. If the real category of the ith sample in the training batch is the same as that of category c, it is taken as 1, otherwise 0, and q_{ic} represents the probability that the ith sample in the predicted training batch belongs to category c.

iv) The calculated loss L back propagation update network weight Θ, as shown in Formula (8):

$$\theta^{'} = \theta - L_r \times \nabla g(L_1) \quad (8)$$

v) The subsequent training batch is chosen to repeat the preceding processes if the traversal is not finished. If the traversal is finished, the training batch satisfaction is assessed and the model weight parameter file for the round is saved.

vi) Repeat the previous stages if the training rounds are not reached; once they are, the model training will be completed.

3.2 Mobile APP design

The Android Integrated Development Environment (Android Studio IDE), which is built on the Windows 10 system platform, is used to create the intrusion plant detector's user interface. It transmits data to the server using the Okhttp network transmission framework and downloads third-party jar development packages, placing them in the lib directory. Following synchronization, it is parsed into a so file before being passed over to the third-party API.

According to Figure 7, the UI interface of the APP uses a bottom navigation bar. The user login main interface, the plant intelligent identification module, the invasive plant question module, and the invasive plant expert identification module are all included in the APP interface design.

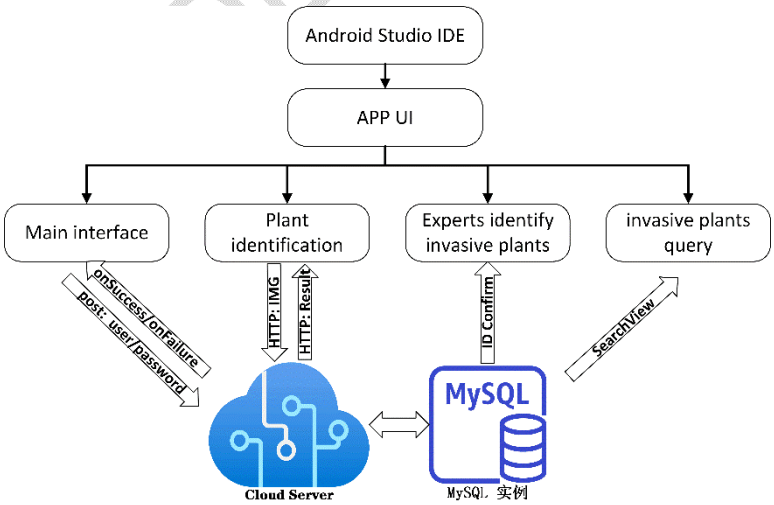

Fig. 7. APP design process

By sending a post request with information parameters such a user name and password, the user login module connects to the server. The login success or failure status is displayed once the server accepts the request and returns callback information onSuccess()/onFailure().

The plant intelligent recognition module offers two image gathering techniques—camera photography and gallery selection—and communicates with the server and mobile device using the HTTP protocol. After receiving the image, the server invokes the plant recognition model to do recognition, and then the client is provided with the recognition result in JSON data format.

The invasive plant information query module writes and implements the invasive plant introduction interface using the SearchView control and the ListView control, and uses the ViewPager component to switch between various View interfaces. It implements the invasive plant interface's search functionality using Filter.

Only users whose identity information is "expert" can identify invasive plants, and the expert identification module of invasive plants primarily uses the MySQL database. The server will keep a record of each user's identification so that others can search and find them.

This paper packages and deploys the program of the agricultural pest recognition model on the remote server of Alibaba Cloud for the deployment of the server-side recognition model. It also adds the mean file, weight file, and label file during the model training process, calls the model, and outputs the recognition. A dll dynamic link library is created once the software has been compiled. The server calls the invasive plant identification model and runs the dll dynamic link file when it receives the plant image sent from the mobile terminal in order to recognize the image that has been uploaded. The submitted photographs and recognition results are kept in the database while the server feeds back the results to the client, making it simple for plant images to be traced in the future.

4. EXPERIMENTAL DETAILS

We conduct an experimental evaluation of the DenseNet-SPP model's performance. It is important to note that after 310,000 training cycles on the MS COCO dataset, we employ the approach of transfer learning to generate weight files to initialize the DenseNet-SPP network weight in order to prevent data overfitting and produce a more robust model.

4.1 Experiment parameter

The training and testing environments of this experiment are the same, both being Windows 10 operating systems, and the Pytorch deep learning framework is used to realize the training and testing process of the complete model. The specific parameters of the experimental environment are presented in Table 1:

Table 1. Benchmark model comparison experiment environment

Test environment	Deploy

CPU	Inter(R) i7-9700 CPU @3.00GHz
GPU	NVIDIA GeForce 1650 4G
RAM	16G
Python	3.8
Pytorch	1.10.1

GoogleNet, ResNet-50, DenseNet-121, DenseNet-SPP (our), and MobilNet-V2 are the five widely used classical convolutional neural network models chosen for the experiment's benchmark model comparison studies. Following that, the improved model in this work is compared to the performance of these five models. Table 2 displays the comprehensive parameter comparison analysis of these five models. The mAP column's value is the detection performance score that was learned using the MS COCO data set [13].

Table 2. Model performance comparison table

Model	Accuracy（top-1）	Accuracy（top-5）	mAP (COCO)	Parement
GoogleNet	77.9%	93.7%	36.8	23.9M
ResNet-101	74.9%	92.1%	40.8	25.6M
DenseNet-121	75.0%	92.3%	39.8	8.1M
MobilNet-V2	71.3%	90.1%	30.6	3.5M

After repeated attempts, the selected training-related hyperparameter values in this experiment are shown in Table 3.

Table 3. Network training hyperparameters

hyperparameters	Accuracy（top-1）
Num Class	8
Optimizer	Adam
Learning rate	0.0001
epochs	200

4.2 Evaluating indicator

Several evaluation indices in the MS COCO data set are frequently utilized in target detection algorithm research to assess the benefits and drawbacks of the algorithm. The most crucial metric is Average Precision (AP), which shows how well a certain kind of target can be detected. Intersection-over Union (IoU), which symbolizes the overlapping of two instance masks, defines AP. There are some additional significant performance metrics, but since the accuracy of target identification is the main concern of this work, other indicators are not contrasted in it. Calculations for AP and mAP are displayed in (9) ~ (12). In order to

avoid affecting the accuracy and recall rate, the assessment index mAP (IoU = 0.50:0.05: 0.95) was chosen.

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{9}$$

$$R = \frac{TP}{TP + FN} \tag{10}$$

$$AP = \int_0^1 \mathrm{P}(R) dR \tag{11}$$

$$\mathrm{m}AP = \frac{mAP_{0.50} + mAP_{0.55} + \cdots + mAP_{0.95}}{10} \tag{12}$$

Among them, TP denotes the proportion of samples for which the model's predicted category matches the actual label category; FP, the proportion of samples for which the model's predicted category does not match the actual label category; and FN, the proportion of samples for which the predicted category was background but the actual label was something else.

4.3 Comparative analysis of the experiment

In order to verify the performance of our proposed invasive plant detection model DenseNet-SPP, we trained these five network models for 200 epochs, and the test accuracy is shown in Table 4:

Table 4. Convolutional Neural Network Model Recognition Performance Comparison Table

Model	AT plant		BP plant		LT plant	
	pre	rec	pre	rec	pre	Rec
GoogleNet	0.423	0.632	0.445	0.453	0.628	0.639
ResNet-101	0.614	0.591	0.615	0.632	0.569	0.689
DenseNet-121	0.853	0.923	0.912	0.935	0.806	0.824
MobilNet-V2	0.837	0.844	0.850	0.726	0.740	0.715
Our	0.872	0.941	0.926	0.881	0.811	0.815

Model	CL plant		AC plant		SC plant	
	pre	rec	pre	rec	pre	Rec
GoogleNet	0.639	0.628	0.776	0.809	0.738	0.776
ResNet-101	0.689	0.569	0.883	0.864	0.861	0.832
DenseNet-121	0.815	0.740	0.811	0.832	0.922	0.911
MobilNet-V2	0.689	0.713	0.753	0.842	0.762	0.755

Our	0.824	0.951	0.915	0.895	0.956	0.953

Model	CG plant		KD plant		other plant	
	pre	rec	Pre	Rec	Pre	Rec
GoogleNet	0.749	0.853	0.829	0.875	0.624	0.618
ResNet-101	0.831	0.824	0.851	0.885	0.639	0.688
DenseNet-121	0.869	0.789	0.907	0.886	0.831	0.811
MobilNet-V2	0.758	0.779	0.828	0.879	0.762	0.759
Our	0.889	0.878	0.929	0.912	0.864	0.786

Table 4 shows that the DenseNet-SPP invasive plant image classification model outperformed other convolutional neural network models in terms of precision rate and recall rate on eight different invasive plant species. This indicates that the DenseNet-The SPP model is better suited for invasive plant image classification tasks than other convolutional neural network models. Our model has the lowest accuracy rate for LT plant species while having the best recognition rate for SC plant species, even reaching 95.6% detection accuracy. Yet, the average identification accuracy of each category is not less than 80%, demonstrating that the DenseNet-based plant recognizer has a significant impact on the data set's noise and plant images with occlusion and other comparable traits. robustness.

In the data set containing invasive plants, the enhanced algorithm is contrasted with other algorithms to confirm its superiority and efficacy. The test results are shown in Table 5. This further demonstrates the efficacy of the approach suggested in this study as the accuracy rate attained on the above is far greater than that of previous benchmark convolutional neural network techniques and the upgraded DenseNet algorithm.

Table 5. Comparison of the average detection accuracy achieved by the algorithm

Model	mAP（%）
GoogleNet	71.02
ResNet-101	83.22
DenseNet-121	86.29
MobilNet-V2	79.63
Our	90.39

4.4 Mobile terminal APP test

An Android-powered phone can have the mobile terminal APP loaded on it. Including the main user login interface, the plant intelligent identification module, the invasive plant query module, and the invasive plant expert identification module, Figure 8 depicts a screenshot of the mobile terminal's APP interface.

(a) APP main interface (b) Plant identification results (c) Experts identify invasive plants (d) invasive plants query

Fig. 8. APP interface of invasive plants identification based on Android mobile phone

The main user interface 9-(a) is displayed when the user launches the APP for the invasive plant identification system. Ordinary users and experts can register separately on the registration interface. The user must upload a professional credential while registering as an expert, and the intrusion can only be performed after the background check. Plant identification: After a successful registration is made in the system, the plant identification module 9-(b) is immediately connected. The user can upload images or take pictures using this interface and upload them to the server, where the DenseNet-SPP model will identify the images and return the results. Figure 8-(d) is the invasive plant query module, and the user can Learn knowledge on this interface. You can also use keyword search to find the details of the invasive plants you want to know. If the user comes across a plant that cannot be identified by the model, he can apply for expert identification. The expert will feedback the identification result on the identification interface, and display 9-(c) on the common user interface. Also, upon testing, the server will utilize the pest identification model and return the results when a user submits it an image of a pest to be recognized. The users' real-time needs can be satisfied in 1 to 2 seconds.

5. CONCLUSION

We created a terminal-oriented invasive plant recognition system based on deep learning technology through the experiment and analysis of this study. To increase the model's robustness and accuracy, the system utilizes the DenseNet network topology and preprocesses and enriches the data set. To prevent over-fitting and achieve a greater recognition accuracy, we apply the proper hyperparameter settings and data improvement techniques during the training phase. The system's average accuracy on the test set was 90.39%. The system also has a main user login interface, invasive plant question and expert identification modules, and a plant intelligent recognition module. Users can use the APP to snap pictures of unknown invasive plants, upload them to the server for plant recognition, and instantly receive the results of that recognition.

REFERENCES

1. Kattenborn, T.; Eichel, J.; Fassnacht, F.E. Convolutional Neural Networks enable efficient, accurate and fine-grained segmentation of plant species and communities from high-resolution UAV imagery. Scientific reports 2019, 9, 17656.
2. Wen, C.; Guyer, D.E.; Li, W. Local feature-based identification and classification for orchard insects. Biosystems engineering 2009, 104, 299-307.
3. Wen, C.; Guyer, D. Image-based orchard insect automated identification and classification method. Computers electronics in agriculture 2012, 89, 110-115.
4. Tan, W.; Zhao, C.; Wu, H. Intelligent alerting for fruit-melon lesion image based on momentum deep learning. Multimedia Tools Applications 2016, 75, 16741-16761.
5. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 2004, 60, 91-110.
6. Dalal, N. Histograms of oriented gradients for human detection. Proc of Cvpr 2005.
7. Kattenborn, T.; Leitloff, J.; Schiefer, F.; Hinz, S. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. ISPRS journal of photogrammetry remote sensing 2021, 173, 24-49.
8. Ray, P.P. Internet of things for smart agriculture: Technologies, practices and future direction. Journal of Ambient Intelligence Smart Environments 2017, 9, 395-420.
9. Bah, M.D.; Hafiane, A.; Canals, R. Deep learning with unsupervised data labeling for weed detection in line crops in UAV images. Remote Sensing in Ecology Conservation 2018, 10, 1690.
10. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017; pp. 4700-4708.
11. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis & Machine Intelligence 2014, 37, 1904-1916.
12. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D. et al. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, 2014; pp. 740-755.
13. Elharrouss, O.; Akbari, Y.; Almaadeed, N.; Al-Maadeed, S. Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches. arXiv preprint arXiv:2206.08016 2022.