Plant Leaf Identification via A Growing Convolution Neural Network with Progressive Sample Learning

Zhong-Qiu Zhao[1,2], Bao-Jian Xie[1], Yiu-ming Cheung[2,4], Xindong Wu[1,3]

[1]College of Computer Science and Information Engineering, Hefei University of Technology, China.
[2]Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China.
[3]Department of Computer Science, University of Vermont, USA.
[4]United International College, Beijing Normal University–Hong Kong Baptist University, Zhuhai, China.
Contact email: z.zhao@hfut.edu.cn

Abstract. Plant identification is an important problem for ecologists, amateur botanists, educators, and so on. Leaf, which can be easily obtained, is usually one of the important factors of plants. In this paper, we propose a growing convolution neural network (GCNN) for plant leaf identification and report the promising results on the ImageCLEF2012 Plant Identification database. The GCNN owns a growing structure which starts training from a simple structure of a single convolution kernel and is gradually added new convolution neurons to. Simultaneously, the growing connection weights are modified until the squared-error achieves the desired result. Moreover, we propose a progressive learning method to determine the number of learning samples, which can further improve the recognition rate. Experiments and analyses show that our proposed GCNN outperforms other state-of-the-art algorithms such as the traditional CNN and the hand-crafted features with SVM classifiers.

1 Introduction

Plant identification is a basic work of plant research and development, and is very important for plant protection and exploration of distinction and genetic relationship between plant species. Usually, the leaves can be easily gotten from plant and have sufficient visible characteristics for differentiating between many species. Currently, plant identification mainly relies on plant scientists with specialized knowledge. However, there exist a large number of plant species, which are hard to be fully identified by a plant scientist. Thus, an automatic plant species identification system by computers and related machine learning algorithms is desired.

The existing state-of-the-art methods of plant leaf identification usually use hand-crafted features such as shape [1], local binary patterns (LBP) [2], pyramid histograms of oriented gradients (PHOG) [3] and their combinations [4], followed

by a trainable classifier such as support vector machine (SVM) [5][6], neural networks [7][8][9], and so on. However, the performances of these methods largely depend on an appropriate set of features, which are varying and need a special design for specific tasks. Moreover, low-level features can be successfully achieved directly, but mid-level and high-level features are difficult to be achieved without any learning procedures [10]. The convolution neural network (CNN) [11] is such a system that can learn generic features for different tasks, directly acting on the two-dimensional image pixels without changing the topology of the input image.

The CNN has received much attention and has been used in lots of applications such as face detection [12], handwriting recognition [13], speech recognition [14], and pedestrian detection [15]. Garcia and Delakis adopted a three-layer CNN structure to detect human face [16], and later they further improved the collecting samples algorithm in the training process [17]. Hinton et al. [18] have proved that increasing the number of feature detectors can significantly improve the performance of the CNNs on various tasks. This particular kind of neural network has no requirement for image preprocessing or feature extraction, and it implements future extraction and pattern classification simultaneously [19].

The CNNs have much fewer parameters to be trained, compared to traditional neural networks. Further, because of the weight sharing technology, the CNNs are easier to be trained. However, there are still some problems in the traditional CNNs. First, the traditional CNNs have fixed structures , which are usually not applicable to all practical problems or the learning of subsequent samples, because the learning of subsequent sample learning often needs to overthrow early learning to establish new weights. Second, it is difficult to determine the number of training samples in the traditional CNNs because too few samples can cause the under-fitting of network learning, while too many may result in over-learning [20].

Addressing the problems mentioned above, this paper proposes a novel approach to construct a growing convolution neural network (GCNN) with a varying architecture for plant leaf identification. The construction and training start with the simplest architecture with a single neuron for each layer, and add new neural cells into each layer, accompanied by modifying the corresponding weights, until the training target is reached. This approach can automatically adjust the CNN structure to fit any specific task. In addition, this paper also proposes a progressive learning method to determine the appropriate number of learning samples for the GCNN. Compared to the traditional CNN, the proposed GCNN model is more effective.

The reminder of the paper is organized as follows. Section 2 reviews the traditional convolution neural network. Section 3 describes the proposed growing convolution neural network. Section 4 presents the details of the progressive sample learning method. Then, Section 5 shows the experimental results and makes some discussions. Finally, a conclusion is drawn in Section 6.

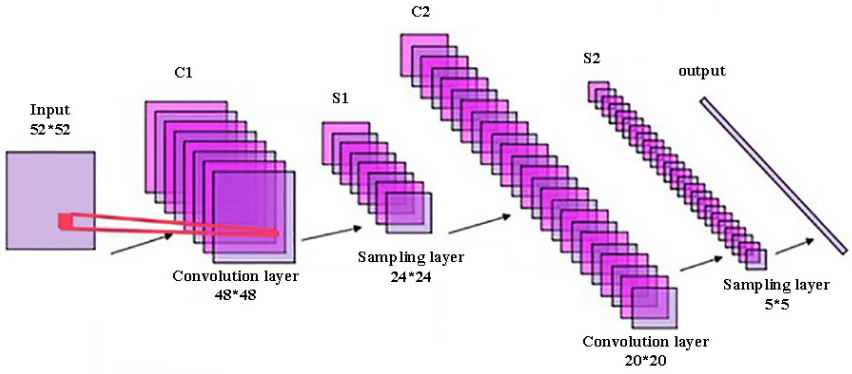

Fig. 1. The structure of a traditional CNN

2.1 The CNN Structure

Figure 1 shows a traditional CNN structure, consisting of an input layer, several convolution layers, several sampling layers, and an output layer. There are unknown weight parameters needed to be trained between two adjacent layers, the feature maps in S2 layer are fully connected to the output layer, otherwise, the feature maps are not fully connected between two layers. Convolution layers are interspersed with sampling layers to reduce computation time and to gradually improve spatial and configural invariance. A sampling layer produces downsampled version of the input maps. At a convolution layer, the features from the previous layer are convolved with learnable kernels and then fed to an activation function to form the output feature maps. In general, the output of convolution layers is computed as follows:

$$x_j^l = f\left(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

where M_j represents the set of input maps, x_j^l is an output map, which is given an additive bias b_j^l , l denotes the layer index, k_{ij}^l denotes convolution kernels, and $'*'$ denotes the convolution operator.

As shown in Figure 1, an input image with simple preprocessing including size normalization (52*52) and graying. Then a convolution operation with a window (5*5) from all directions extracts the features of the input image, and there by features are obtained at the C1 layer. The sampling layer S1 produces downsampled version of the input to eliminate the deviation and image distortion. The size of feature map (48*48) in C1 becomes (24*24) by sampling

operation of S1, and the size of sampling window of S1 is 2*2. The output of the sampling layers is computed as follows.

$$x_j^l = f\left(\frac{1}{n}\sum_{i\in M_j} x_i^{l-1} + b^l\right) \tag{2}$$

where n is the size of window from the convolution layer to the sampling layer so that the output image is n-times smaller along both spatial dimensions.

2.2 The Connections and Weights of The Network

In traditional neural networks, there is an unknown parameter with respect to each connection. However, the CNN uses a weight-sharing technology. Units at a layer are organized in planes, sharing the same set of weights. This technology can greatly reduce the number of free parameters and can be used to detect the representation of the same characteristics in different angles. For the example in Figure 1, each unit in one feature map has 25 inputs connected to a 5*5 area in one feature map of the convolution layer. This unit is called the receptive field [21]. Therefore, there are 25 tunable weights and a trainable bias for each unit. Layer C1 is a convolutional layer with six feature maps, and the size of the feature maps of C1 is 48*48. So there are $6 * 26 = 156$ tunable parameters and $26*6*48*48 = 359,424$ connections between the input and C1 layers. Layer S1 is a sampling layer with the sampling window of (2*2), which produces six feature maps of (24*24). Four pixels in C1 are sampled into one pixel in S1, and there is one tunable weight and a tunable bias for each feature map between C1 and S1. So there are $2*6 = 12$ tunable parameters and $6*[(2*2)*(24*24)+(24*24)] = 17280$ connections between C1 and S1. Similarly, there are $16*26 = 416$ tunable parameters and $26 * 16 * 24 * 24 = 166400$ connections between S1 and C2, $2 * 16 = 32$ tunable parameters and $16 * [(4 * 4) * (5 * 5) + (5 * 5)] = 6800$ connections between C2 and S2, and $16 * 5 * 5 + 5 = 2005$ tunable parameters and $16 * 5 * 5 + 5 = 2005$ connections between S2 and the output layer. So the network in Figure 1 totally contains 551,909 connections, but only 2,621 free parameters need learning. The tunable parameters are updated by iteration based on the back-propagation (BP) algorithm [22].

3 The Proposed Growing Convolution Neural Network (GCNN)

The performance of CNN depends greatly on the number of neurons. The recognition rate of the network increases with the growth of the number of neurons, but the computation cost also increases accordingly. Current researches mainly rely on prior knowledge to design the network structure. In this paper, we propose a growth algorithm to construct the CNN, in which the network grows up itself until it solves the target problem. Thereby, the best tradeoff can be achieved between classification accuracy and computation cost.

3.1 Initialization of The Network

The initial network structure is set very simple, as shown in Figure 2. The initial network consists of two layers of convolution (C1,C2) and two layers of sampling (S1,S2), each of which contains two feature maps. The initial network consists of two branches (branch 1 and branch 2). There are 367 parameters in the convolution network 'Net0' needed to be trained. The weights are updated by the output and the corresponding errors at previous iteration, and the weight

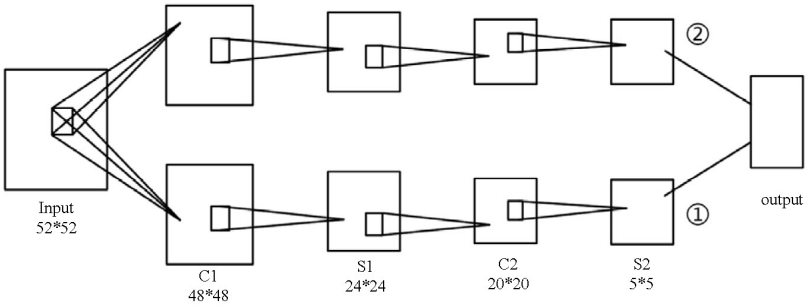

Fig. 2. The initial convolution neural network 'Net0'

3.2 The First Round of Growth

Branch 3 sprouts at the first round of grows on the basis of network 'Net0', which results in the network 'Net1' shown in Figure 3. The growth rule is as follows. The two feature maps contained in the sampling layer S1 are combined to constitute the third feature map in the convolution layer C2, and the number of feature maps in the convolution layer C2 and sampling layer S2 feature map is increased by one to three. The feature map combination can increase the diversity of features. We initialize the new neurons while the weights in Branches 1 and 2 remain unchanged. The learning route only goes through the output layer and Branch 3 at the new growth round. The connection weights of Branch 3 are modified iteratively until the overall error of the network converges to a small value. If the error convergence speed becomes smaller than a threshold value, then the training process enters the second round of growth.

3.3 The Second Round of Growth

The second growth of the network adds a feature map to each layer, producing Branch 4 as shown in Figure 4. Thereby, there are three feature maps in the C1 or

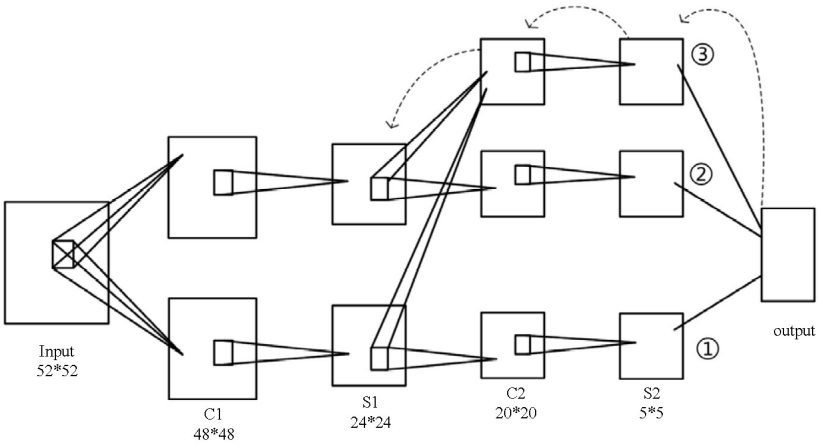

Fig. 3. The first round of growth: the network 'Net1'

S1 layer, while four in the C2 or S2 layer. All learning outcomes of network Net1 are kept unchanged, and the weight modification by back-propagation algorithm only involves Branch 4. The weights of Branch 4 are updated until the overall error of the network converges to a small value. If the error convergence speed becomes smaller than a threshold value, then the training process continues to grow according to the following algorithm.

3.4 The Whole Growth Algorithm and Back-propagation Algorithm

The whole growing algorithm of the GCNN is shown in Algorithm 1. And Figure 5 shows an example of a maturing GCNN network with six branches.

Back-propagation Algorithm

Feedforward Pass Let l denote the layer index, and then the output of the lth layer is defined as [23]:

$$x^l = f\left(W^l x^{l-1} + b^l\right) \tag{3}$$

where $f(\cdot)$ is an activation function, which is usually set to be sigmoid function or hyperbolic tangent function. The outcomes of layer $l-1$, namely x^{l-1}, pass forward the activation function to produce the final output. Given N training samples $\{x_n\}_{n=1,\ldots,N}$ and the corresponding target vectors $\{t_n\}_{n=1,\ldots,N}$, for a

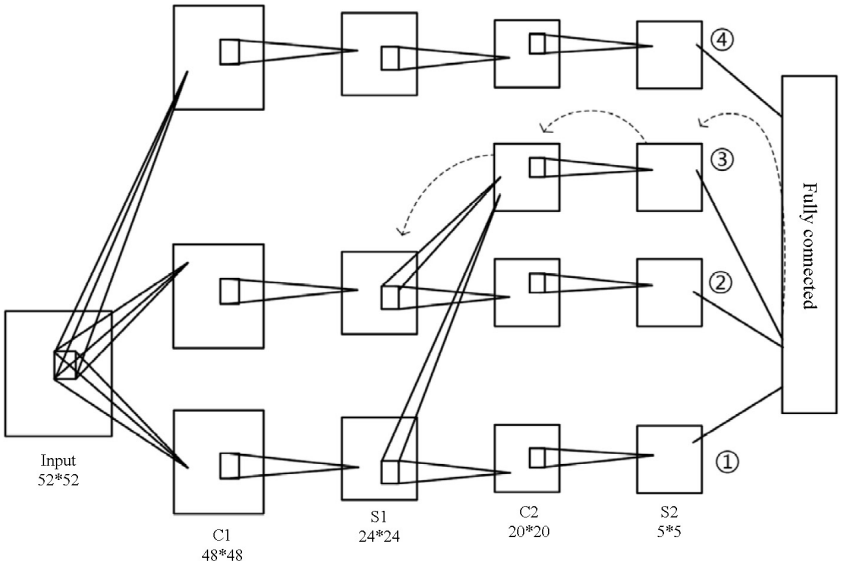

Fig. 4. The second round of growth: the network 'Net2'

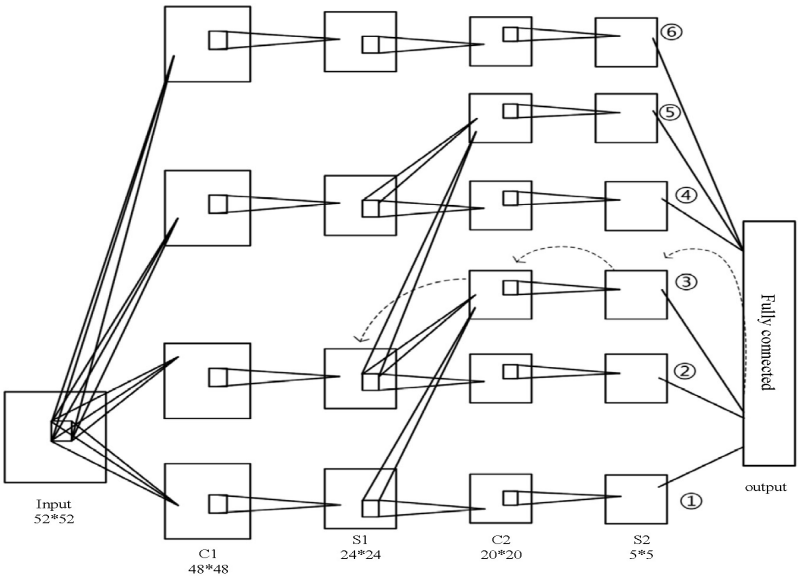

Fig. 5. An example of the CNN network structure of six branches

Algorithm 1 The Whole Growing Algorithm

SET:
E denotes the squared error for training samples;
$E0$ denotes the squared error threshold;
s denotes the convergence speed of BP;
$s0$ denotes the convergence speed threshold;
Nb denotes the number of current network branches;
Initialize: $E0$, $s0$, η(the learning rate of BP), $Net0$;

```
if E > E0 then
    (continue to grow the network:)
    if s > s0 then
        train the network with the BP algorithm;
    else
        (add a new branch to the current network:)
        if Nb is an even number then
            add a new branch starting from the S1 layer, and the feature maps in S1
            layer are merged as a new starting point of the new branch;
            Nb = Nb + 1;
        else
            add a new branch starting from the input layer;
            Nb = Nb + 1;
        end if;
    end if;
else
    stop growing;
end if
```

Output: A matured CNN.

multiclass classification problem with c classes, we have the following squared-error loss function:

$$E^N = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}(t_k^n - y_k^n)^2 \tag{4}$$

where y^n is the output with respect to the input vector x_n. In training process, the targets of multiclass problems are organized as "one-of-c" codes where the kth element of t_n is positive if the pattern x_n belongs to class k.

Backpropagation Pass The errors propagate backward through the network, which can be considered as 'sensitivities' of each unit with respect to perturbations of the bias b. The derivative of the error can be defined

$$\delta = \frac{\partial E}{\partial b} \tag{5}$$

It is this derivative that is backpropagated from higher layers to lower layers using the following propagation rule:

$$\delta^l = \left(W^{l+1}\right)^T \delta^{l+1} \cdot f'\left(W^l x^{l-1} + b^l\right) \tag{6}$$

Finally, the delta rule is used to update the weight of a given neuron. For the layer of l, the delta of the error with respect to each weight of this layer is computed by the vector of inputs and the vector of sensitivities:

$$\frac{\partial E}{\partial W^l} = x^{l-1} \left(\delta^l\right)^T \tag{7}$$

$$\Delta W^l = -\eta \frac{\partial E}{\partial W^l} \tag{8}$$

where η is the learning rate.

4 The Progressive Sample Learning Method

It is difficult to determine the proper number of training samples for traditional CNNs. Too few can lead to under-fitting of network learning, while too many may result in over-learning. Therefore, we propose a progressive sample learning method (PSLM) which is self-organizing. The method starts with learning a small amount of samples from each class and then adds to the training set the samples from the classes whose errors are larger until the whole squared-error is satisfactory. This method not only reduces the number of training samples and simplifies learning, but also allows adding new categories. The details of the PSLM is shown in Algorithm 2. The misclassification rate threshold in this algorithm is set as $\varepsilon = 0.3$.

Algorithm 2 The Progressive Sample Learning Method

SET:
E denotes the squared error for training samples;
$E0$ denotes the squared error threshold;
A_i denotes the current training set for class i;
A_i' denotes an additional small sample set from class i excluding A_i;
e_i denotes the misclassification rate for class i;
ε denotes a misclassification rate threshold.

while $E > E0$ **do**
  Train the CNN using the training set $\bigcup_i A_i$;
  Compute e_i of the CNN for all i;
  for each i **do**
    if $e_i > \varepsilon$ **then**
      $A_i = A_i \bigcup A_i'$;
    else
      $A_i = A_i$;
    end if
  end for
end while

Output: A learned CNN.

5 Experiments and Analyses

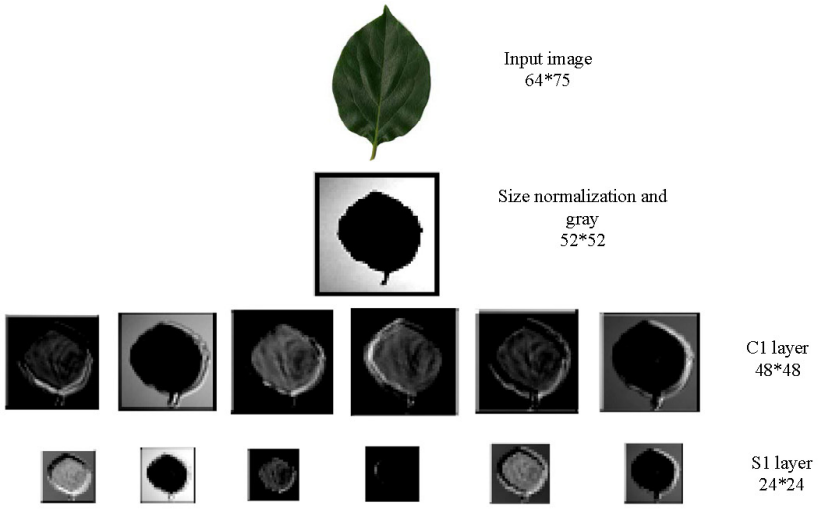

Fig. 6. An example of 2D feature maps of a plant leaf image in the CNN

The experiments includes four parts: (1)The traditional CNNs to classify plant leaf images; (2)The CNN with growing structure to classify plant leaf images; (3)The CNN with growing structure and progressive sample learning method to classify leaf images; (4)other state-of-the-art methods such as HSV+SVM, Phog+SVM, HSV+Phog+SVM. We evaluate various algorithms on the dataset of the ImageCLEF2012 [24] Plant Identification task. The database comes from the ImageCLEF2012 competition, and it contains 126 tree species in the French Mediterranean area, and the images for each species are contributed by various people, but taken from the same individual plant. The database consists of three kinds of images: scans, scan-like photos (called pseudo-scans) and natural photos. We select 40 classes from ImageCLEF2012 database, each of which contains 60 leaf images of scans and scan-like for evaluations. And 40 images of each class are selected for training and the remaining for test. The evaluation indexes include MCR (misclassification rate), recognition time, RSME (root mean square error), and recognition rate. We set the sample average error threshold $E0 = 0.2$, the predetermined threshold of error convergence speed $s0 = 0.05$ and the learning rate $\eta = 0.0002$. All experiments are implemented by the MATLAB7.1.0.183 (R14) Service Pack 3 running on a computer with the CPU of 8 Quad-Core AMD, memory of 4 194 304 kB, and the WIN7 OS.

Figure 6 shows an example of 2D feature maps of a plant leaf image in the CNN. Figure 7 shows the RMSE and MCR variations of the growing CNN by iteration, respectivesly, from which we can see that the RMSE and MCR

both decrease by iteration. Table 1 shows the comparison of recognition rate and recognition time between different algorithms. And the number of unknown parameters needed learning for each CNN network is also listed in Table 1.

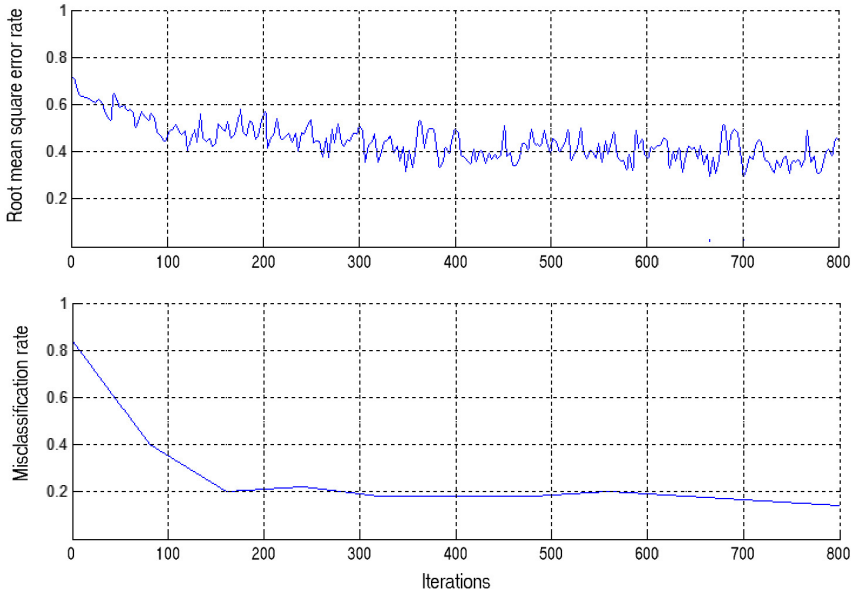

Fig. 7. The MCR and RMSE variations of the CNN of 7 branches in training process

From Table 1, we can see that traditional CNN performs better than any single feature with an SVM classifier, but has no advantages over fusion of several features such as HSV and Phog. The recognition rate of the GCNN increases with the expansion of the network scale, but the recognition time increases at the same time. When the CNN grows to 9 branches, the recognition rate reaches 87.94%, and the PLSM helps to further improve the recognition rate, which reaches 88.14%. However, for the CNN of 7 branches, the recognition rate already reaches 86.42%, and 87.22% with the PLSM method, which is significantly higher than that of traditional CNN. And the recognition time of the CNN of 7 branches is much less than that of 9 branches. So the CNN of 7 branches may be a good choice, considering the tradeoff between the recognition rate and recognition time in real applications. In addition, the results in Table 1 also show that our proposed GCNN involves much fewer unknown parameters needed learning than the traditional CNN, which indicates that the GCNN owns simpler structure and is easier to train.

Table 1. The comparison of recognition rate and recognition time between different algorithms

Algorithms	Recognition Rate (%)	Recognition Time Per Image (Second)	Number of Parameters in Network
CNN	80.34	0.53	2621
CNN+PLSM	82.05	0.51	2621
HSV+SVM	65.11	0.29	–
Phog+SVM	78.10	0.31	–
HSV+Phog+SVM	83.04	0.41	–
GCNN (5 branches)	66.53	0.43	906
GCNN (5 branches)+PLSM	71.85	0.41	906
GCNN (6 branches)	79.71	0.49	1087
GCNN (6 branches)+PLSM	81.06	0.48	1087
GCNN (7 branches)	86.42	0.61	1266
GCNN (7 branches)+PLSM	87.22	0.60	1266
GCNN (8 branches)	87.68	0.78	1447
GCNN (8 branches)+PLSM	87.89	0.78	1447
GCNN (9 branches)	87.94	0.95	1626
GCNN (9 branches)+PLSM	88.14	0.95	1626

6 Conclusion

In this paper, we have proposed GCNN with a variable topology structure, and evaluated it on plant leaf recognition. The GCNN automatically grows to a proper size according to the complexity of specific classification task. Also, we have presented a progressive sample learning algorithm for the GCNN, which can automatically determine the proper number of training samples, avoiding under-learning or over-fitting. The experiments and analyses on the ImageCLEF2012 plant Identification Dataset show that our proposed GCNN outperforms the other state-of-the-art algorithms such as traditional CNN and hand-crafted features with SVM classifiers.

7 Acknowledgments

This research was supported by the National Natural Science Foundation of China (Nos. 61375047 and 61272366), the 973 Program of China (No.2013CB329604), the 863 Program of China (No.2012AA011005), the Program for Changjiang Scholars and Innovative Research Team in University of the Ministry of Education of China (No. IRT13059), the US National Science Foundation (NSF CCF-0905337), the Faculty Research Grant of Hong Kong Baptist University (No. FRG2/12-13/082), the Hong Kong Scholars Program (No.XJ2012012), China Postdoctoral Science Foundation (No. 2013M540510), and the Fundamental Research Funds for the Central Universities of China.

References

1. Arora, A., Gupta, A., Bagmar, N., Mishra, S., Bhattacharya, A.: A plant identification system using shape and morphological features on segmented leaflets: Team iitk, clef 2012. In: CLEF (Online Working Notes/Labs/Workshop). (2012)
2. Ren, X.M., Wang, X.F., Zhao, Y.: An efficient multi-scale overlapped block lbp approach for leaf image recognition. In: Intelligent Computing Theories and Applications. Springer (2012) 237–243
3. Chen, J., Bai, Y.: Classification of smile expression using hybrid phog and gabor features. In: Computer Application and System Modeling (ICCASM). Volume 12., IEEE (2010) V12–417
4. Ma, L.H., Zhao, Z.Q., Wang, J.: Apleafis: an android-based plant leaf identification system. In: Intelligent Computing Theories. Springer (2013) 106–111
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20** (1995) 273–297
6. Sun, B.Y., Huang, D.S., Guo, L., Zhao, Z.Q.: Support vector machine committee for classification. In: Advances in Neural Networks–ISNN 2004. Springer (2004) 648–653
7. Zhao, Z.Q., Huang, D.S.: A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability. Applied Mathematical Modelling **31** (2007) 1271–1281
8. Zhao, Z.Q., Huang, D.S., Sun, B.Y.: Human face recognition based on multi-features using neural networks committee. Pattern Recognition Letters **25** (2004) 1351–1358
9. Zhao, Z.Q., Gao, J., Glotin, H., Wu, X.: A matrix modular neural network based on task decomposition with subspace division by adaptive affinity propagation clustering. Applied Mathematical Modelling **34** (2010) 3884–3895
10. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM (2009) 609–616
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
12. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks **8** (1997) 98–113

13. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: 2013 12th International Conference on Document Analysis and Recognition. Volume 2., IEEE Computer Society (2003) 958–958
14. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine **29** (2012) 82–97
15. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2013) 3626–3633
16. Garcia, C., Delakis, M.: A neural architecture for fast and robust face detection. In: 16th International Conference on Pattern Recognition, 2002. Proceedings. Volume 2., IEEE (2002) 44–47
17. Garcia, C., Delakis, M.: Convolutional face finder: A neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004) 1408–1423
18. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
19. Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07, IEEE (2007) 1–8
20. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding **106** (2007) 59–70
21. Theunissen, F.E., Sen, K., Doupe, A.J.: Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds. The Journal of Neuroscience **20** (2000) 2315–2331
22. Zhang, J.R., Zhang, J., Lok, T.M., Lyu, M.R.: A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Applied Mathematics and Computation **185** (2007) 1026–1037
23. Bouvrie, J.: Notes on convolutional neural networks. (2006)
24. Zheng, P., Zhao, Z.Q., Glotin, H.: Zhaohfut at imageclef 2012 plant identification task. In: CLEF (Online Working Notes/Labs/Workshop), Citeseer (2012)