Neurocomputing 235 (2017) 228–235

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Plant identification using deep neural networks via optimization of transfer learning parameters

Mostafa Mehdipour Ghazi[a,*], Berrin Yanikoglu[a], Erchan Aptoula[b]

[a] Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey
[b] Institute of Information Technologies, Gebze Technical University, Kocaeli, Turkey

ARTICLE INFO

Communicated by Zidong Wang

Keywords:
Convolutional neural networks
Deep learning
Plant identification
Transfer learning
Inverse rank score

ABSTRACT

We use deep convolutional neural networks to identify the plant species captured in a photograph and evaluate different factors affecting the performance of these networks. Three powerful and popular deep learning architectures, namely GoogLeNet, AlexNet, and VGGNet, are used for this purpose. Transfer learning is used to fine-tune the pre-trained models, using the plant task datasets of LifeCLEF 2015. To decrease the chance of overfitting, data augmentation techniques are applied based on image transforms such as rotation, translation, reflection, and scaling. Furthermore, the networks' parameters are adjusted and different classifiers are fused to improve overall performance. Our best combined system has achieved an overall accuracy of 80% on the validation set and an overall inverse rank score of 0.752 on the official test set. A comparison of our results against the results of the LifeCLEF 2015 plant identification campaign shows that we have improved the overall validation accuracy of the top system by 15% points and its overall inverse rank score on the test set by 0.1 while outperforming the top three competition participants in all categories. The system recently obtained a very close second place in the PlantCLEF 2016.

1. Introduction

Automated plant identification is concerned with the classification of photographed plant organs into botanical species, using machine learning algorithms. The problem faces a number of challenges due to inter-class similarities within a plant family and large intra-class variations in background, occlusion, pose, color, and illumination.

Deep learning architectures have been popular and able to achieve significant success in many problems in recent years [1–4]. Contrary to traditional machine learning methods in which features are chosen manually and extracted through instructed algorithms, deep learning networks automatically discover increasingly higher level features from data [5]. These networks, through utilization of large amounts of data and exploitation of parallel architectures with high-performance computing techniques, are able to overcome obstacles that were previously associated with shallow networks.

Training deep neural networks that contain millions of parameters requires massive amounts of data in the order of millions of samples, in order to properly constrain the optimization. The need for large amounts of data and high computational resources to train these networks from scratch steer researchers to adapt pre-trained networks to a desired task domain by means of fine-tuning with the domain-specific data [6–8]. This approach is a form of *transfer learning*, which aims to apply knowledge learned from a problem to another, different, but related problem [9,10].

There are two ways to apply transfer learning with deep networks. One possibility is to utilize the pre-trained network with the learned weights to obtain features that would be subsequently used in the new problem. Here, the outputs of the network, prior to the last fully-connected layer, constitute features of interest. Another option is to *fine-tune* the network weights by training the network with the new dataset. In this case, the number of nodes at the output layer must be modified to match the number of classes in the new problem. Furthermore, in both scenarios, the data must match the input size of the pre-trained network.

Choosing between these two options depends on the size and similarity of the new dataset to the original one used for pre-training. In general, fine-tuning the entire network is only suggested when the new dataset is large enough; otherwise, there would be concerns about overfitting especially within the earlier layers of the network. Since these layers extract generic, low-level features such as edges and color blobs, they do not change significantly and can be utilized for several visual recognition tasks [11]. In contrast, the later layers of the network gradually become tuned to the details in the problem and extract high-

* Corresponding author.
E-mail addresses: mehdipour@sabanciuniv.edu (M. Mehdipour Ghazi), berrin@sabanciuniv.edu (B. Yanikoglu), eaptoula@gtu.edu.tr (E. Aptoula).

http://dx.doi.org/10.1016/j.neucom.2017.01.018
Received 11 April 2016; Received in revised form 5 August 2016; Accepted 7 January 2017
Available online 11 January 2017
0925-2312/

level features of the data. These layers are thus problem-dependent and more likely to change during fine-tuning as they are closer to the output layers which receive larger back-propagated error differentials.

In this paper, we describe a state-of-the-art plant identification system based on deep learning, for recognizing the plant species in a given unconstrained photograph. For this purpose, we fine-tune pre-trained deep convolutional neural networks of AlexNet [3], GoogLeNet [12], and VGGNet [13] using the LifeCLEF 2015 plant task dataset [14].

Our contribution is two-fold. First, we have developed a state-of-the-art solution for automated plant identification from unconstrained plant photographs with validation accuracies of 76.87% and 78.44% using individually optimized and fine-tuned models of GoogLeNet and VGGNet, respectively, and an overall accuracy of 80.18% using a score level fusion of these two deep networks. The resulting system improves the validation accuracy of the winning plant identification system of LifeCLEF 2015 by a wide margin of 15% points and its inverse rank score by 0.1. Also, a slightly more fine-tuned version of this system ranked a very close second in the main category of PlantCLEF 2016, where the identification is based on multiple images, and the first in the single-image based identification [15].

Second, beyond a mere application of deep learning to plant recognition, we provide an in-depth performance evaluation of the critical factors affecting the fine-tuning of pre-trained models; specifically iteration size, batch size, and data augmentation. Our findings determine the relative significance of each of the aforementioned variables on performance, thus paving the way for more optimal utilization of valuable computational resources. We believe this would be of interest to researchers who choose to fine-tune pre-trained systems for other similar problems.

The rest of this paper is organized as follows. Section 2 starts with an overview of related work and is followed by Section 3 which covers details of the three deep learning models under study, i.e. AlexNet, GoogLeNet, and VGGNet. Section 4 presents the proposed method for data augmentation, fusion, and parameter adjustment using deep learning models. Section 5 is dedicated to the conducted experiments and their results. Finally, Section 6 wraps up the paper with a discussion of our major findings.

2. Related work

Up until about a decade ago, the majority of earlier studies on plant identification from photographs were mostly concerned with acquisition, preprocessing, feature extraction, and supervised learning from isolated leaf images [16]. Since color information is not a highly discriminative feature in this context, the proposed methods for leaf recognition have frequently utilized shape analysis [17–22], texture analysis [23–26], and venation analysis [27–29]. In addition to leaf recognition, a number of studies in that period have addressed the problem of identifying plants from flower images [30,31].

The early years of the Conference and Labs of the Evaluation Forum's (CLEF) plant identification contests had been rather focused on the feasibility of the problem at hand, with limited datasets under study. Following their success, the goal shifted rapidly towards fully automatic plant identification from unconstrained photographs, requiring high levels of robustness against varying illumination conditions, viewpoints, scale, and capacity to deal with a variety of plant organs. These campaigns, that have been organized as ImageCLEF [32] and LifeCLEF [33], are devoted to promoting and evaluating multilingual and multimodal information retrieval systems. Plant identification campaigns held by CLEF is the most well-known annual events that benchmark content-based identification of plant species using real-world plant datasets–including images from leaves, branches, stems, flowers, and fruits–with a large number of samples.

During the early CLEF campaigns between 2011 and 2013, the submitted systems applied a variety of generic feature extraction and classification approaches, typically using Support Vector Machines and Random Forests as classifiers. The steady but slow performance progress changed drastically in 2014 when participants started exploring deep neural networks. The winner of the LifeCLEF 2014 plant identification task [34] was the only participant in that campaign who utilized a deep learning based method, training AlexNet [3] from scratch to classify 500 plant species. This method was able to outperform all the fellow participants with a great margin. Following this successful achievement and the new developments in deep learning, the 2015 campaign was performance-wise dominated by deep learning based systems [35–37] that relied on fine-tuning the pre-trained GoogLeNet model [12] against 1,000 species.

A look at the results obtained by the different methods submitted to the plant identification tasks of CLEF campaigns immediately reveals that fine-tuning deep learning methods clearly outperforms the previously employed strategies. This observation on the fine-tuning of deep neural networks is by no means limited to the problem of plant identification. For example, [6] transferred the learned weights of AlexNet from object recognition to scene recognition using a dataset of over seven million labeled scene images. Likewise, the approach in [7] fine-tuned AlexNet weights to extract features useful for categorizing urban tribes and [8] retrained GoogLeNet and VGGNet to design very deep, two-stream networks for action recognition in videos.

3. Deep learning models

We have evaluated the performance of the following three powerful architectures of deep neural networks for the plant identification problem.

3.1. AlexNet

The model proposed in [3] is a deep convolutional neural network successfully trained on roughly 1.2 million labeled images of 1,000 different categories from the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) dataset [38]. As the winner of ILSVRC 2012, the AlexNet architecture has about 650,000 neurons and 60 million parameters. It includes five convolutional layers, two normalization layers, three max-pooling layers, three fully-connected layers, and a linear layer with Softmax activation in the output. Moreover, it uses the dropout regularization method [39] to reduce overfitting in the fully-connected layers and applies Rectified Linear Units (ReLUs) for the activation of those and the convolutional layers.

3.2. GoogLeNet

Inspired by the network-in-network approach [40], GoogLeNet is an inception architecture [12] that combines the multi-scale idea and dimension reduction layers based on the Hebbian principle and embedding learning. This enables one to increase the width and depth of the network for an improved generalization capacity per a constant computational complexity.

More specifically, GoogLeNet, the winner of ILSVRC 2014, possesses roughly 6.8 million parameters with nine inception modules, two convolutional layers, one convolutional layer for dimension reduction, two normalization layers, four max-pooling layers, one average pooling, one fully-connected layer, and a linear layer with Softmax activation in the output. Each inception module in turn contains two convolutional layers, four convolutional layers for dimension reduction, and one max-pooling layer. GoogLeNet also uses dropout regularization in the fully-connected layer and applies the ReLU activation function in all of the convolutional layers.

During training, GoogLeNet connects two auxiliary classifiers to the intermediate layers of the network to effectively perform backpropagation through all the layers. In other words, this network adds auxiliary losses to the total loss to make the earlier stages more discriminative

and increase the backpropagated gradient. The additional classifiers include one average pooling layer, one convolutional layer for dimension reduction, two fully-connected layers, and a linear layer with Softmax activation.

3.3. VGGNet

As the first runner-up in ILSVRC 2014, VGGNet [13] uses a homogeneous architecture to investigate the effects of increased convolutional network depth on performance. The largest VGGNet architecture involves 144 million parameters from 16 convolutional layers with very small receptive fields (3 × 3), five max-pooling layers of size 2 × 2, three fully-connected layers, and a linear layer with Softmax activation in the output. This model also uses dropout regularization in the fully-connected layer and applies ReLU activation to all the convolutional layers.

However, despite its satisfactory performance in multiple transfer learning tasks, this network has a greater number of parameters compared to AlexNet and GoogLeNet, which makes it computationally more expensive to evaluate and requires a considerable amount of memory in optimizing the learning parameters.

4. The proposed method

Deep neural networks are generally trained based on the prediction loss minimization. Let **x** and **y** be the input (images) and corresponding output (class labels), the objective of the training is to iteratively minimize the average loss (empirical risk) defined as

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(f(w; x_i), y_i) + \lambda R(w), \tag{1}$$

where N is the number of data instances (mini-batch) in every iteration, L is the loss function, f is the predicted output of the network depending on the current weights w, and R is the regularization term or weight decay with the Lagrange multiplier λ.

We use the stochastic gradient descent (SGD) which is a highly common optimization algorithm utilized in deep networks to update the weights.

$$w_{t+1} = \mu w_t - \alpha \nabla J(w_t), \tag{2}$$

where μ is the momentum weight for the current weights w_t and α is the learning rate. The network weights are randomly initialized during training the network from scratch, while in fine-tuning deep models, the network weights are initially set to the pre-trained network weights.

The principal goal of transfer learning with deep networks is to effectively exploit learned weights with very large datasets of generic object recognition tasks to solve the new problem. To this end, we compare fine-tuning deep networks with training them from scratch while tuning all important adjustable parameters of the networks for performance maximization Section 4.1.

Data augmentation is realized in both the training and testing phases as explained in Section 4.2, and classification outputs for the augmented images in the test phase are combined following the procedure described in Section 4.3. Finally, individually optimized deep neural networks are fused in order to enhance the overall system accuracy.

The training and test subsets of the LifeCLEF 2015 plant identification dataset are used to train and test the utilized deep networks, respectively. Details of this dataset, as the largest available plant image collection, are explained in Section 5.1.

4.1. Network parameters

We evaluate the importance of iteration size, batch size, and the amount of data augmentation on the performance of the utilized deep networks, while other network parameters are kept at fixed values. As all three of these parameters directly affect a network's training time, the goal was to study their relative importance, as well as to obtain the best system for the plant identification problem. While the answers are directly relevant to the plant identification problem, we hope that the relative effectiveness of these modifications can shed some light on where to concentrate in transfer learning with deep neural networks.

The number of iterations indicates the number of weight updates performed on the whole network; this parameter is varied from 100,000 to 500,000 for fine-tuning and set to 200,000 for training from scratch. The batch size refers to the number of images in each mini-batch considered to estimate the gradient before each weight update. In this work, we increment the number of images in each batch among 20, 40, and 60. The number of patches indicate the number of image patches extracted from the original image, so as to constrain the weight values and reduce overfitting. The value of this parameter is set to either 10 or 80 throughout our experiments.

Generally speaking, these parameters are inter-linked and any changes in one parameter can affect the others. However, since training deep networks takes a long time even with parallel architectures, we analyze the effects of each parameter one by one.

4.2. Data transformation and augmentation

Given the large size and high architectural complexity of the utilized deep convolutional neural networks, the need to artificially expand our utilized dataset is evident as it both minimizes the possibility of overfitting and maximizes the benefits of fine-tuning. To accomplish this goal, we use a variety of label-preserving image transforms, namely translation, reflection, rotation, and scaling.

The implemented data augmentation steps are shown in Fig. 1. Since cropping and reflecting are automatically realized through prevalent deep learning frameworks such as Caffe ([41]), the main data augmentation approach employed in this work is based on extracting and scaling random square patches from the original image and augmenting them with image rotation. The left sub-block of Fig. 1 depicts the proposed steps based on rotation and scaling while the right sub-block shows the proposed transforms of [3] and can be performed via Caffe.

Assuming an input image of size $M \times N$ pixels, K random square patches of length min(M,N) are extracted from the original image around its center. Meanwhile, the original image is rotated both

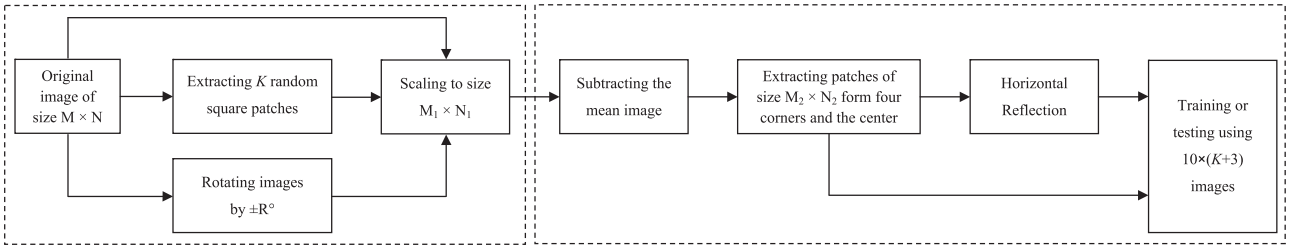

Fig. 1. Block diagram of the proposed method for data augmentation. The right sub-block can be performed via deep learning frameworks such as Caffe and the left one can be applied for rotation and scaling with square patches while preserving aspect ratio.

clockwise and counterclockwise by a small degree of R, resulting in two more patches. Next, the largest possible square image is cropped from the center of each rotated image to preserve the original aspect ratio. All the extracted $K + 2$ patches as well as the original image are then scaled to a fixed size of $M_1 \times N_1$ pixels and the mean image is subtracted from them. Finally, given the input size of the networks, five patches of size $M_2 \times N_2$ pixels are extracted from four corners and the center of each image and reflected horizontally, giving $10 \times (K + 3)$ images in total.

We implemented two different sets of scenarios. In the first one, we used only the original image in the left sub-block, resulting in 10 patches in total (the top arrow in the left sub-block in Fig. 1). In the second scenario, we set K =5 and R =10 degrees, resulting in a total of 80 patches, as shown in the left sub-block.

As required by [3], M_1 and N_1 are assumed to be equal to 256 and M_2 and N_2 are set to the expected input size of the respective networks, i.e. 224 × 224 for GoogLeNet and VGGNet, and 227 × 227 for AlexNet.

4.3. Score-based fusion

The score-based fusion approach is employed for combining the classification results of the patches, as well as later combining outputs of different classifiers assigned to deep learning models. To combine the classification results obtained for different patches within a single network, we use the sum rule to fuse the prediction scores assigned to each class for all the augmented patches. Later on, the prediction scores obtained from every network classifier are combined using the same sum rule fusion to achieve the final prediction.

5. Performance analysis

In this section, we explain the dataset used in the experiments and discuss the experimental results.

5.1. The dataset

We used the plant task dataset of LifeCLEF 2015 [14] to evaluate the performance of the deep neural networks explored in this work. The large training dataset contains 91,758 labeled images of different plant organs (e.g. flowers, fruits, leaves, and stems), from 1,000 species of trees, herbs, and ferns. The testing set contains 21,446 images with recently released ground-truth labels. These datasets are very challenging since different photographers have collected photographs from different locations using varying conditions of background, pose, color, and illumination.

To validate our results, we used proportionate stratified random sampling and divided the training dataset into two subsets of training and validation. Specifically, for each plant category, we randomly selected one-fifth of the available samples from each individual class as samples of the validation set. Therefore, the obtained training and validation sets include 70,904 and 20,854 images, respectively.

Table 1 shows the number of total instances and classes for each category used for experiments. The *Entire* column in the table is a category in which the photographed images contain pictures of different parts of plants, while the *Overall* column refers to all images within the dataset.

5.2. Results

We used the Caffe deep learning framework ([41]) with the implemented and pre-trained models of AlexNet, GoogLeNet, and VGGNet obtained from the Caffe Model Zoo provided by the Berkeley Vision and Learning Center (BVLC). These models are trained on the ILSVRC 2012 dataset with 1.2 million labeled images of 1,000 classes. The fine-tuned versions of these models constitute the baselines used in Sections 5.2.1 and 5.2.2.

5.2.1. Training from scratch vs. fine-tuning

To observe the effects of transfer learning on system performance, we evaluated the performance of fine-tuning the aforementioned pre-trained models with the training subset of LifeCLEF 2015 until 100,000 iterations and compared them against training the same networks from scratch until 200,000 iterations. The batch size for each network was set to 20 while the weight decay factor and learning rate were initialized to 0.0002 and 0.001, respectively. Moreover, the step size for updating the learning rate was set to 12,000 iterations with a coefficient–gamma–of 0.96.

Table 2 shows the accuracies obtained by the selected Caffe models on the validation set. When considering the results of training the models from scratch, AlexNet has performed the best with an overall validation accuracy of 44.19% and closer to its own fine-tuned model (49.63%). We conclude that if deep learning networks need to be trained from scratch, architectures with relatively smaller number of adjustable weights may perform better, especially with a relatively small amount of data.

Moreover, comparing the difference between fine-tuning and training from scratch in this problem, we see that fine-tuning has clearly resulted in higher values of validation accuracy compared to training the models from scratch.

Also, we notice that fine-tuning performance of these three networks is directly related with their relative sizes, with the most successful one being VGGNet. In fact, VGGNet has achieved the best results in almost all categories–except for the Flower category where GoogleLeNet has obtained the best results with an overall accuracy of 61.93%.

Since fine-tuning the GoogLeNet and VGGNet models has achieved the best results, we have conducted the later experiments with these two systems, especially with GoogLeNet since VGGNet is much slower to train.

5.2.2. Effects of number of iterations

We fine-tuned the pre-trained models until 500,000 iterations while recording the values of training loss and validation set accuracies. Figs. 2 and 3 show the variations of training loss and the validation accuracy against iterations during the fine-tuning of the utilized deep networks, respectively. As it can be seen in Fig. 3, validation accuracy rises rapidly in the earlier iterations and increases slowly afterwards for all three networks. In line with results reported in Tables 2 and 3, the performance of VGGNet and GoogLeNet are highly similar and both superior to that of AlexNet.

The improvement in performance is significant for all the evaluated networks as the number of iterations increases from 100,000 to 500,000. This shows that these networks are resilient to overfitting

Table 1
Details of the utilized dataset for plant identification.

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
Number of training samples	6,447	12,567	21,531	6,072	10,367	9,576	4,344	70,904
Number of validation samples	1,683	3,668	6,694	1,648	3,000	3,029	1,132	20,854
Number of testing samples	2,088	6,113	8,327	1,423	2,690	221	584	21,446
Number of species	891	993	967	755	899	351	649	1,000

Table 2
Classification accuracies (%) for different training scenarios. Bold values indicate the best results in each category.

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
AlexNet from scratch (200 K, 20, 10x))	23.53	27.32	45.49	35.50	33.87	92.11	33.66	44.19
AlexNet fine-tuning (100 K, 20, 10x)	28.64	30.10	53.45	43.99	45.03	90.99	31.18	49.63
GoogLeNet from scratch (200 K, 20, 10x)	17.35	16.90	36.41	24.82	20.63	83.13	13.16	33.78
GoogLeNet fine-tuning (100 K, 20, 10x)	44.09	38.36	**67.93**	57.65	60.57	94.16	37.01	61.06
VGGNet from scratch (200 K, 20, 10x)	16.93	16.47	32.12	22.69	20.23	81.91	13.43	31.90
VGGNet fine-tuning (100 K, 20, 10x)	**44.68**	**42.58**	66.37	**59.59**	**61.90**	**95.05**	**38.87**	**61.93**

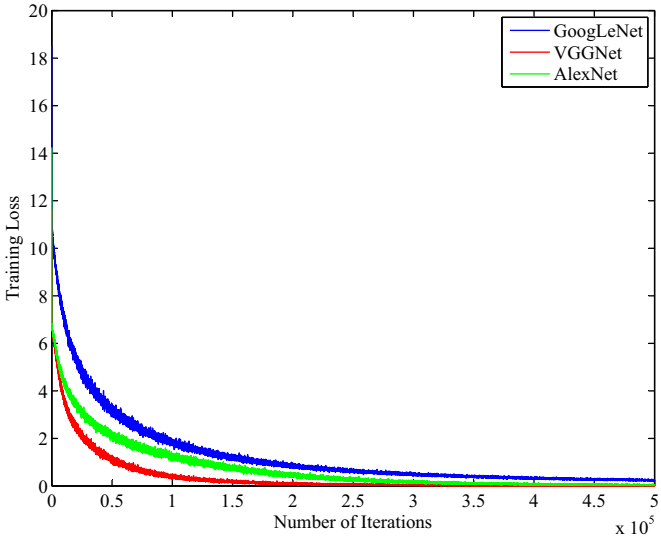

Fig. 2. Effects of increasing the number of iterations on the training loss.

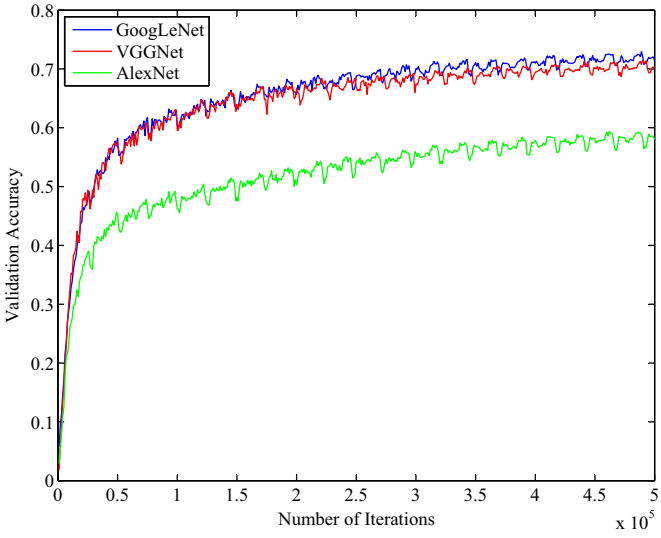

Fig. 3. Effects of increasing the number of iterations on the overall accuracy of the validation set.

due to their careful architecture design and the utilized data augmentation scheme.

5.2.3. Effects of batch size

To see the effects of batch size on the performance of the fine-tuned GoogLeNet, we evaluated the performance with batch sizes ranging from 20 to 60. The results presented in the first, fourth, and fifth rows of Table 3 show that increasing the batch size from 20 to 60 improves the overall accuracy of this model from 61.06% to 65.18%.

However, a larger batch size increases the training duration. For instance, training with a tripled batch size takes the same amount of time as training with the triple number of iterations.

Consequently, since accuracy improves more significantly with the increased number of iterations rather than a larger batch size, it seems more reasonable to use a smaller but sufficient batch size, together with a higher number of iterations to train deep networks.

5.2.4. Effects of data augmentation

We evaluated the performance of GoogleLeNet with 10 or 80 patches extracted from the input image as described in Section 4.2. At the testing phase, we applied the score-based fusion technique to combine all predictions before reaching the final decision. Comparing these two cases, we see that using 8-fold data augmentation increases the validation set accuracy significantly, i.e. from 61.06% to 67.32%.

5.2.5. Fine-tuning best networks

To further improve performance, we evaluated GoogLeNet with the combined adjustments; specifically with 80 patches for data augmentation, 300,000 iterations, and a batch size of 60. We also explored obtaining the best VGGNet model using augmented data with 80 patches until 500,000 iterations. As shown in Table 3, using the best combined parameters with the longest training durations, these two networks improve their best performance obtained thus far by more than 7% points, achieving 76.87% and 78.44%, respectively.

5.2.6. Discussion

Based on the results shown in Table 3, we conclude that increasing the number of iterations, the number of patches used for data augmentation, and the batch size, improves baseline performance–with the number of iterations having the greatest and the batch size having the least significant effect. As an example, increasing the batch size from 20 to 60 increases the training time 3-fold, but does not match the performance obtained with increasing the number of iterations by the same amount, i.e. 65.18% versus 68.57%, from 61.06%.

While we have not tried higher values for the attempted parameters, we believe that the performance increase would continue with all three, with the most pronounced effects expected from data augmentation. Furthermore, as discussed in Section 5.2.5, the combination of the best parameters brings in further improvements, but as the goal of this paper is to study their effect in isolation, we did not run prohibitively long experiments with all combinations.

5.2.7. Overall results with classifier fusion

We finally combined the best models of GoogLeNet and VGGNet using the score-based fusion technique and achieved an overall accuracy of 80.18%, as shown in the last row of Table 3. Comparing this result with the validation results of the winner of LifeCLEF 2015 plant identification campaign [35] reveals that we have been able to improve their performance by a considerable amount of 14.99% points.

As our last experiment, we evaluated the above fusion system using the recently released *test set* of LifeCLEF 2015 ([33]). To make the

Table 3
Classification accuracies (%) with adjusted parameters and data augmentation during fine-tuning of the pre-trained models. Bold values indicate the best obtained results in each category. The values inside parenthesis indicate the number of iterations, batch size, and number of patches for data augmentation, respectively.

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
GoogLeNet (100 K, 20, 10x)	44.09	38.36	67.93	57.65	60.57	94.16	37.01	61.06
GoogLeNet (300 K, 20, 10x)	55.08	46.05	76.23	67.96	68.07	95.77	45.76	68.57
GoogLeNet (500 K, 20, 10x)	55.02	47.66	76.43	69.11	69.13	95.58	45.49	69.11
GoogLeNet (100 K, 40, 10x)	45.63	41.03	70.05	61.23	60.73	93.20	36.49	62.48
GoogLeNet (100 K, 60, 10x)	50.98	41.09	73.07	63.59	65.50	94.19	41.52	65.18
GoogLeNet (100 K, 20, 80x)	54.01	48.06	73.38	64.99	68.63	94.85	39.84	67.32
GoogLeNet (300 K, 60, 80x) (*)	67.56	60.28	83.30	76.88	76.30	96.93	54.33	76.87
VGGNet (100 K, 20, 10x)	44.68	42.58	66.37	59.59	61.90	95.05	38.87	61.93
VGGNet (500 K, 20, 10x)	56.33	52.81	77.53	68.93	70.47	97.29	51.68	71.24
VGGNet (500 K, 20, 80x) (**)	68.09	64.37	84.40	77.00	76.67	97.92	58.75	78.44
GoogLeNet (*) & VGGNet (**)	**71.24**	**65.16**	**86.90**	**79.13**	**78.93**	**98.02**	**59.45**	**80.18**

results fully comparable with those obtained by task participants, we used the user-based inverse rank score provided by the campaign ([33]). The average inverse rank score is defined as follows

$$S = \frac{1}{U}\sum_{u=1}^{U}\frac{1}{P_u}\sum_{p=1}^{P_u}\frac{1}{N_{u,p}}\sum_{n=1}^{N_{u,p}} s_{u,p,n} \quad (3)$$

where U is the number of users collecting the query images; P_u is the number of individual plants observed by the u-th user; $N_{u,p}$ is the number of pictures taken from the p-th plant observed by the u-th user; and $s_{u,p,n}$ is the inverse of the rank of the correct species for the given image, ranging from 0 to 1. The results of this experiment are compared to the best methods submitted to the plant identification task of LifeCLEF 2015 and shown in Table 4. As it can be seen, our proposed method significantly outperforms all other systems in all categories with an overall inverse rank score of 0.752.

5.3. Time complexity of different networks

We measured the complexity of the utilized deep neural networks in terms of the training time per iteration during the forward-backward pass and the testing time per image during feature extraction and classification. Table 5 shows the average training time per iteration with a batch size of 20 and the average testing time per image by fusing the classification results for 10 extracted patches. All the approaches were implemented in MATLAB, Python, and Caffe under Linux and all experiments were run on a Tesla K40c-based system with 12 GB of video memory.

As shown in Table 5, AlexNet has the lowest training and testing time among the three networks while GoogLeNet's training and testing durations are 1.5 times longer than those of AlexNet. VGGNet, on the other hand, is almost four times slower than GoogLeNet in both training and testing.

In order to demonstrate the capability of our network as a powerful low-level and high-level feature extractor, we illustrate the network layer weights learned from the retrained GoogLeNet model on the LifeCLEF 2015 plant dataset in Fig. 4. As the figure shows, the first layer weights are tuned to extract edges or color blobs but the higher levels are more likely to extract specific patterns seen in plant organs.

Table 5
Average training time per iteration and testing time per image, in milliseconds.

	AlexNet	GoogLeNet	VGGNet
Training time	303	447	1,787
Testing time	21	31	114

6. Conclusions

We applied transfer learning to identify plant species using deep convolutional neural networks. The utilized networks are based on pre-trained deep learning models of AlexNet, GoogLeNet, and VGGNet. We evaluated these networks on the plant task datasets of LifeCLEF 2015, with different network parameters and data augmentation, and finally combining the best classifiers' predictions to improve overall system performance. The results show that our best combined system has surpassed the overall validation accuracy of [35], the winner of the LifeCLEF 2015 plant identification campaign, by almost 15% points and its overall inverse score obtained on the test set by 0.1 while outperforming the top three systems in all categories.

Comparing the relative performance of these networks reveals that fine-tuning GoogLeNet and VGGNet results in obtaining higher performances compared to fine-tuning AlexNet, with the best accuracy being obtained by VGGNet with 78.44%. On the other hand, training AlexNet from scratch outperforms GoogLeNet and VGGNet probably due to AlexNet's simpler architecture. Therefore, we can conclude that although we benefit from transfer learning, training from scratch using simpler networks gives the opportunity to define novel and computationally efficient networks.

Our findings indicate that the most significant factor affecting fine-tuning performance is the number of iterations while data augmentation comes second. On the other hand, while increasing the batch size improves accuracy, we observed that increasing the number of iterations is a better use of computation time, considering the time complexity versus performance improvements. We hope and believe that the observations collected in this work will shed some light to other similar visual recognition problems.

Table 4
Average inverse rank score for the best plant identification systems evaluated on the test set of LifeCLEF 2015.

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
SNUMedinfo ([35])	0.498	0.531	0.784	0.602	0.600	0.766	0.326	0.652
QUT RV ([36])	0.416	0.448	0.738	0.558	0.524	0.694	0.291	0.590
Inria Zenith ([37])	0.398	0.453	0.723	0.559	0.501	0.713	0.302	0.581
Proposed Method	**0.666**	**0.624**	**0.861**	**0.723**	**0.745**	**0.848**	**0.429**	**0.752**

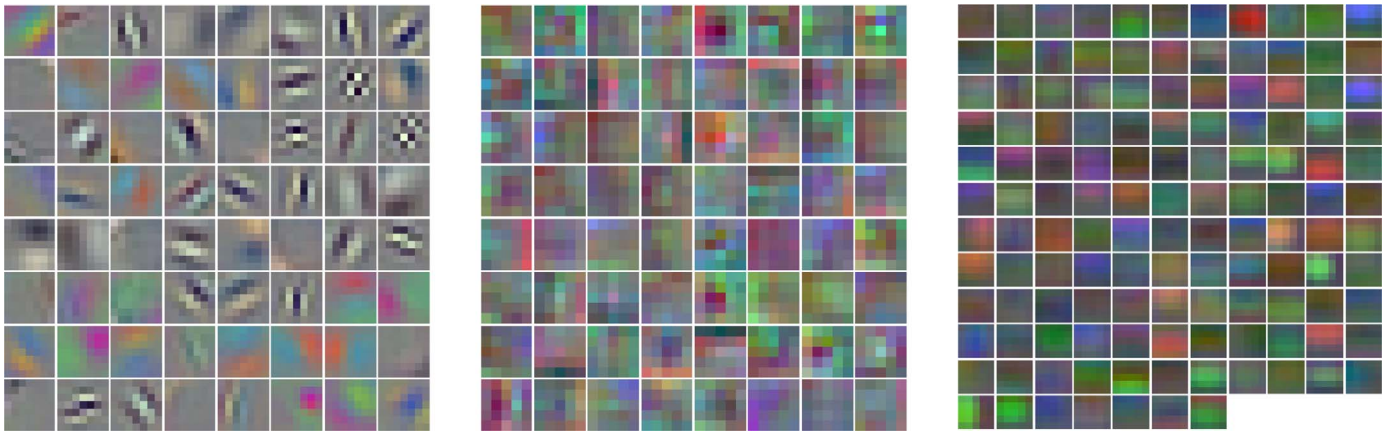

Fig. 4. The network filter banks of the first layer (*Left*), the middle layer (*Center*), and the last layer (*Right*) learned by fine-tuning GoogLeNet using the LifeCLEF 2015 plant dataset.

Acknowledgements

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the grant number 113E499.

References

[1] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554. http://dx.doi.org/10.1162/neco.2006.18.7.1527.

[2] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, Y. LeCun, Learning convolutional feature hierarchies for visual recognition, in: Advances in neural information processing systems, 2010, pp. 1090–1098.

[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in neural information processing systems (NIPS), 2012, pp. 1106–1114.

[4] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929. http://dx.doi.org/10.1109/tpami.2012.231.

[5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. http://dx.doi.org/10.1038/nature14539.

[6] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning deep features for scene recognition using places database, in: Advances in neural information processing systems, 2014, pp. 487–495.

[7] Y. Wang, G.W. Cottrell, Bikers are like tobacco shops, formal dressers are like suits: Recognizing urban tribes with Caffe, in: IEEE Winter Conference on Applications of Computer Vision, 2015. http://dx.doi.org/10.1109/wacv.2015.121.

[8] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, Towards good practices for very deep two-stream convnets, Computing Research Repository (CoRR) arXiv:1507.02159.

[9] L. Torrey, J. Shavlik, Transfer learning, in: Machine Learning Applications and Trends, 2009, pp. 242–264. http://dx.doi.org/10.4018/978-1-60566-766-9.ch011.

[10] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359. http://dx.doi.org/10.1109/TKDE.2009.191.

[11] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015. http://dx.doi.org/10.1109/cvpr.2015.7298594.

[13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, Computing Research Repository (CoRR) arXiv:1409.1556.

[14] A. Joly, H. Goëau, C. Spampinato, P. Bonnet, W.-P. Vellinga, R. Planqué, A. Rauber, S. Palazzo, B. Fisher, H. Müller, LifeCLEF 2015: multimedia life species identification challenges, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction, 2015, pp. 462–483. http://dx.doi.org/10.1007/978-3-319-24027-5_46.

[15] ImageCLEF / LifeCLEF - Multimedia Retrieval in CLEF, ⟨http://www.imageclef.org/lifeclef/2016/plant⟩, PlantCLEF 2016.

[16] J.S. Cope, D.P.A. Corney, J.Y. Clark, P. Remagnino, P. Wilkin, Plant species identification using digital morphometrics: a review, Expert Syst. Appl. 39 (8) (2012) 7562–7573. http://dx.doi.org/10.1016/j.eswa.2012.01.073.

[17] D. Guyer, G. Miles, M. Schreiber, O. Mitchell, V. Vanderbilt, Machine vision and image processing for plant identification, Trans. ASAE 29 (6) (1986) 1500–1507. http://dx.doi.org/10.13031/2013.30344.

[18] C.-L. Lee, S.-Y. Chen, Classification of leaf images, Int. J. Imaging Syst. Technol. 16 (1) (2006) 15–23. http://dx.doi.org/10.1002/ima.20063.

[19] J.C. Neto, G.E. Meyer, D.D. Jones, A.K. Samal, Plant species identification using elliptic fourier leaf shape analysis, Comput. Electron. Agric. 50 (2) (2006) 121–134. http://dx.doi.org/10.1016/j.compag.2005.09.004.

[20] J.-X. Du, X. Wang, G.-J. Zhang, Leaf shape based plant species recognition, Appl. Math. Comput. 185 (2) (2007) 883–893. http://dx.doi.org/10.1016/j.amc.2006.07.072.

[21] F.-Y. Lin, C.-H. Zheng, X.-F. Wang, Q.-K. Man, Multiple classification of plant leaves based on gabor transform and LBP operator, in: Advanced Intelligent Computing Theories and Applications, 2008, pp. 432–439. http://dx.doi.org/10.1007/978-3-540-85930-7_55.

[22] O.M. Bruno, R. de Oliveira Plotze, M. Falvo, M. de Castro, Fractal dimension applied to plant identification, Inf. Sci. 178 (12) (2008) 2722–2733. http://dx.doi.org/10.1016/j.ins.2008.01.023.

[23] T. Beghin, J.S. Cope, P. Remagnino, S. Barman, Shape and texture based plant leaf classification, in: Advanced Concepts for Intelligent Vision Systems, 2010, pp. 345–353. http://dx.doi.org/10.1007/978-3-642-17691-3_32.

[24] A.N. Hussein, S. Mashohor, M.I. Saripan, A texture-based approach for content based image retrieval system for plant leaves images, in: IEEE 7th International Colloquium on Signal Processing and its Applications, 2011, pp. 11–14. http://dx.doi.org/10.1109/cspa.2011.5759833.

[25] E. Aptoula, B. Yanikoglu, Morphological features for leaf based plant recognition, in: 2013 IEEE International Conference on Image Processing, 2013, pp. 1496–1499. http://dx.doi.org/10.1109/icip.2013.6738307.

[26] B. Yanikoglu, E. Aptoula, C. Tirkaz, Automatic plant identification from photographs, Mach. Vis. Appl. 25 (6) (2014) 1369–1383. http://dx.doi.org/10.1007/s00138-014-0612-7.

[27] J. Clarke, S. Barman, P. Remagnino, K. Bailey, D. Kirkup, S. Mayo, P. Wilkin, Venation pattern analysis of leaf images, in: Advances in Visual Computing, 2006, pp. 427–436. http://dx.doi.org/10.1007/11919629_44.

[28] Y. Li, Z. Chi, D.D. Feng, Leaf vein extraction using independent component analysis, in: IEEE International Conference on Systems, Man, and Cybernetics, 2006, pp. 3890–3894. http://dx.doi.org/10.1109/ICSMC.2006.384738.

[29] J.S. Cope, P. Remagnino, S. Barman, P. Wilkin, The extraction of venation from leaf images by evolved vein classifiers and ant colony algorithms, in: Advanced Concepts for Intelligent Vision Systems, 2010, pp. 135–144. http://dx.doi.org/10.1007/978-3-642-17688-3_14.

[30] A.-X. Hong, G. Chen, J. li Li, Z. ru Chi, D. Zhang, A flower image retrieval method based on ROI feature, J. Zhejiang Univ. Sci. 5 (7) (2004) 764–772. http://dx.doi.org/10.1631/jzus.2004.0764.

[31] M.-E. Nilsback, A. Zisserman, Delving deeper into the whorl of flower segmentation, Image Vis. Comput. 28 (6) (2010) 1049–1062. http://dx.doi.org/10.1016/j.imavis.2009.10.001.

[32] H. Goëau, P. Bonnet, A. Joly, V. Bakic, D. Barthelemy, N. Boujemaa, J.-F. Molino, The ImageCLEF 2013 plant identification task, in: CLEF (Working Notes), 2013.

[33] H. Goëau, P. Bonnet, A. Joly, LifeCLEF plant identification task 2015, in: CLEF (Working Notes), 2015.

[34] Q. Chen, M. Abedini, R. Garnavi, X. Liang, IBM research Australia at LifeCLEF 2014: Plant identification task, in: CLEF (Working Notes), 2014.

[35] S. Choi, Plant identification with deep convolutional neural network: SNUMedinfo at LifeCLEF plant identification task 2015, in: CLEF (Working Notes), 2015.

[36] Z. Ge, C. McCool, C. Sanderson, P. Corke, Content specific feature learning for fine-grained plant classification, in: CLEF (Working Notes), 2015.

[37] J. Champ, T. Lorieul, M. Servajean, A. Joly, A comparative study of fine-grained classification methods in the context of the LifeCLEF plant identification challenge 2015, in: CLEF (Working Notes), 2015.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y.

[39] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. (2014) 1929–1958.

[40] M. Lin, Q. Chen, S. Yan, Network in network, Computing Research Repository (CoRR) arXiv:1312.4400.

[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678. http://dx.doi.org/10.1145/2647868.2654889.

Mostafa Mehdipour Ghazi is a researcher at the Computer Vision and Pattern Analysis Laboratory at Sabanci University, Istanbul, Turkey. He has obtained his graduate degrees in Electrical and Electronics Engineering from University of Tehran and Sabanci University with high honors. He is currently working on developing computer vision and deep learning algorithms for object recognition, face recognition, and plant identification. His main research interests are statistical image processing and analysis, computer vision, machine learning, deep learning, pattern recognition and data mining.

Berrin Yanikoglu received a double major in Computer Science and Mathematics from Bogazici University and her Ph.D. degree in Computer Science from Dartmouth College, USA in 1988 and 1993, respectively. She is currently a Professor at Sabanci University, Istanbul, Turkey. Dr. Yanikoglu's research interests lie in the areas of pattern recognition and machine learning with applications to image understanding, currently focusing on handwriting recognition, biometric verification, and plant image recognition. She received first place positions in several international competitions in signature verification and plant image recognition along with students and colleagues.

Erchan Aptoula received the B.Sc. degree in Computer Engineering from Galatasaray University, Istanbul, Turkey in 2004 and the M.Sc. and Ph.D. degrees in Computer Science from Louis Pasteur University, Strasbourg, France in 2005 and 2008, respectively. He is currently an Associate Professor at the Institute of Information Technologies of Gebze Technical University at Kocaeli, Turkey, working on color and multivariate mathematical morphology and hyper-spectral image analysis as well as on content-based image description and retrieval.