STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 1, 2019
DOI: 10.24193/subbi.2019.1.01

TRIST: TREE RECOGNITION INTELLIGENT SYSTEM

LAURA ONAC

Abstract. Plant recognition represents a challenging computer vision problem due to the great variations of shape and texture among plant organs, within the same species. This paper proposes a light-weight, but reasonably deep Convolutional Neural Network architecture able to carry out this classification task. Multiple experiments were conducted with the proposed network architecture on the MEW2012 and Swedish leaf datasets. The experiments showed promising results, outperforming the current state-of-the-art systems that rely exclusively on a convolutional network for plant classification.

1. Introduction

Plant recognition has always been a challenging computer vision problem because of the great variations of shape and texture among plant organs, within the same species. This paper focuses on the classification of trees based on their leaves and describes a supervised deep learning technique, namely a Convolutional Neural Network (CNN), able to carry out this task.

The traditional approach to image pattern recognition has been to extract hand-crafted features from the images and then train a certain classifier with the resulting feature vectors. So, the performance of the systems employing this approach depends heavily on the underlying predefined features. In the plant identification literature, there are some common leaf features used for classification, such as circularity, eccentricity, roundness, aspect ratio [4]. However, the feature engineering process is quite complex and tedious, always needing extensive revision when changing the dataset.

But, over the past few years, Convolutional Neural Networks have become extremely popular in the field of computer vision. They are able to automatically detect important features such as shape or texture, without human

Received by the editors: April 26, 2018.

2010 *Mathematics Subject Classification.* 92B20.

1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: Artificial Intelligence – *Learning*.

Key words and phrases. machine learning, computer vision, convolutional neural networks, image classification, leaf recognition.

assistance in the feature design process. Earlier studies showed that pure CNN approaches [1, 6, 7] seem more powerful than other classifiers trained with previously extracted features [2, 10, 13, 16, 18].

The CNN architecture described in this paper was tested first on the MNIST [11] benchmark dataset, and later on the MEW2012 [13] and Swedish [14] leaf datasets. The architecture showed promising results, which outperform the current state-of-the-art systems that rely exclusively on a convolutional network for plant classification.

Problem description and relevance. Plant recognition implies the use of leaves, flowers, bark or a combination between them, in order to identify the corresponding species. As a result, a lot of research papers were published on this topic, such as [1, 2, 4, 6, 7, 10, 12, 13, 14, 16, 17, 18]. Plant identification is also quite a difficult task for humans to accomplish because it requires domain-specific expertise, as well as experience. Nonetheless, recognizing plants, but particularly trees, holds great importance to mankind as they provide essential resources such as oxygen, food, medicine and wood. Many professions (e.g. environmental science, forestry, landscape architecture) involve the correct identification of trees and would greatly benefit from an automated system which can achieve this task.

2. Background

Convolutional neural networks [11] are a type of deep, feed-forward artificial neural networks that have proven to be greatly effective in image recognition tasks. They are able to recognize certain visual patterns directly from pixel images with minimal preprocessing. In order to ensure some degree of shift, scale and distortion invariance, convolutional networks employ a combination of three architectural elements: local receptive fields, shared weights and sub-sampling. They manage to incorporate knowledge about the invariances of 2D shapes by using local connection patterns, therefore forcing the extraction of local features.

Like any other feed-forward network, a CNN is composed of an input layer, one or more hidden layers, and an output layer. But, as opposed to ordinary networks which only contain fully-connected layers, CNNs incorporate some additional types, namely convolutional and pooling layers.

Each layer transforms a volume of activations into another, through a differentiable function. The last fully-connected layer is called the output layer and in classification settings it represents the class scores. So, the entire network transforms the input image, layer by layer, from the original pixel values to the final class scores.

The Convolutional layer (CONV) [9] is the core building block of a Convolutional network, as it does most of the computational heavy lifting. It computes the output of neurons that are connected to local regions in the input. The output represents the dot product between the neuron's weights and the corresponding elements in a small region from the input. This layer's parameters are actually a set of learnable, small 2-dimensional filters (or kernels). During the forward pass, each filter is slid (or convolved) across the width and height of the input and the dot products between the entries of the filter and the input at the corresponding positions is computed. The result will be a 2-dimensional activation map that gives the responses of that filter at every spatial position.

The size of the feature maps, which are the result of the convolution, is controlled by using two techniques: stride and zero-padding [9]. In this work, our goal is to preserve the size of the input image and we achieve this by combining the zero-padding with a stride equal to 1.

Pooling layers (POOL) [9] are periodically inserted in-between successive convolutional layers. Pooling is a form of non-linear down-sampling and these layers progressively reduce the spatial size of the representation. Thus, pooling layers reduce the amount of parameters and computation in the network. They most commonly use a *max* or *average* operation in order to resize the input. This layer does not have any trainable parameters.

Finally, after several Convolutional and Pooling layers, the high-level reasoning in the network relies on the Fully-connected layers (FC) [9]. The output from the Convolutional and Pooling layers represent high-level features of the input image. The purpose of the Fully-connected layer is to use these features to classify the input image into various classes based on the training dataset.

Dropout layers (DROP) [15] are used in order to avoid overfitting the training dataset. On those layers, units and their connections are randomly dropped during the training process.

Each neuron in the Convolutional and Fully-connected layers has an activation function, also known as a non-linearity, which is applied element-wise on the neuron's output. *Leaky ReLU* is such a function. It is very similar to the ReLU (Rectified Linear Unit) [5] activation function, but it attempts to fix the *dying ReLU* problem [8]. Instead of the function being zero when the input is negative, a leaky ReLU will instead have a small negative slope. This function is defined in Formula 1 [8]:

$$f(x) = max(\varepsilon x, x), \tag{1}$$

where x is the output of the neuron and ε is a small constant.

Another activation function, often used on the last Fully-connected layer of a network-based classifier is called Softmax [9]. The Softmax function squashes the outputs of each unit to be between 0 and 1, and it also divides each output such that the total sum of the outputs is equal to 1. So, the output of this function is equivalent to a categorical probability distribution. Softmax is defined in Formula 2 [9]:

$$\sigma(Z_j) = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}}, \tag{2}$$

where Z is the input vector, j is the index of the output unit and K is the number of output units.

Training a neural network essentially means using the training set to adjust the weights and biases in the network's layers, in order to improve its performance. The training process is an iterative one and it is composed of two major steps: forward propagation and backward propagation. The forward pass implies getting an image from the input layer, through the hidden layers and to the output layer. The obtained result is then compared to the expected output and, during the backward pass, the error for each unit is computed and its parameters are updated accordingly.

Improving performance usually means minimizing the loss function, which in this case is *cross entropy* [3]. Cross entropy indicates the distance between what the model believes the output distribution should be and what the original distribution really is. It is defined in Formula 3 [3]:

$$H(y, p) = -\sum_{i} y_i log(p_i), \tag{3}$$

where y is the expected vector and p is the predicted one.

3. Literature Review

Over the years, there have been several studies carried out on automatic plant recognition. Most of them perform the classification task based on the plants' leaves. As opposed to flowers, leaves are available over a longer period of time and in a much greater number. Also, leaves are more specific to a certain tree than bark is.

What follows is a presentation of other works in the literature that focus on plant recognition. For the purposes of this paper, the related studies will

be divided into those that use purely CNNs and those that employ different classification methods.

The classification of plants using purely CNNs was only recently proposed by authors such as Jassman [7] in 2015, He et al. [6] and Atabay [1], both in 2016.

Jassman [7] aims to develop a system capable of classifying images with natural background. The five networks described in the paper are trained using a dataset created by the author, which contains 15 species of plants with 30 samples each. The best network obtained a top-1 accuracy of 56.66%.

He et al. [6] proposed four different CNN architectures trained with 20 species from the ICL dataset. They started with a CNN with two convolutional layers, two average pooling layers, followed by a fully-connected single-layer perceptron. The first variation introduced was dropout in the last layer, then they added a single connected layer between the second and third layer, and at last, they developed an advanced version of the single connected layer. With the fourth version, they achieved a precision of 91.90%.

Atabay [1] developed two convolutional networks with the same architecture, but different activation functions. For one, he used ReLU, and for the other, ELU. His networks were trained with the Flavia [18] and Swedish [14] leaf datasets, obtaining 97.24% and 99.11% mean average precision on the validation sets, respectively. Those results were achieved using ELU activation.

Traditional methods for leaf recognition generally imply the extraction of features to be used subsequently as input for a classifier. For the feature extraction phase, a variety of techniques were employed: Novotný & Suk [13] used Fourier descriptors, Sulc & Matas [16] developed a method using multi-scale histograms, Çuğu et al. [2] produced some hand-crafted features, Wu et al. [18] extracted digital morphological features, and Kumar et al. [10] used histograms of curvature over scale. The most popular classifiers are Support Vector Machines [16, 2] and the Nearest Neighbor algorithm [13, 10]. Çuğu et al. [2] obtained the final result by merging the results of the SVM with those of a CNN. Wu et al. [18] used a Probabilistic Neural Network as the classifier.

4. TRIST-Net: Methodology

The architecture of the Convolutional Neural Network is illustrated in Figure 1 and it is composed of 10 layers: $INPUT \rightarrow CONV1 \rightarrow POOL1 \rightarrow CONV2 \rightarrow POOL2 \rightarrow CONV3 \rightarrow POOL3 \rightarrow FC \rightarrow DROP \rightarrow OUTPUT$. This architecture was chosen due to the promising results obtained in experiments conducted first on the MNIST dataset [11], and later on the MEW2012 [13] and Swedish [14] datasets.

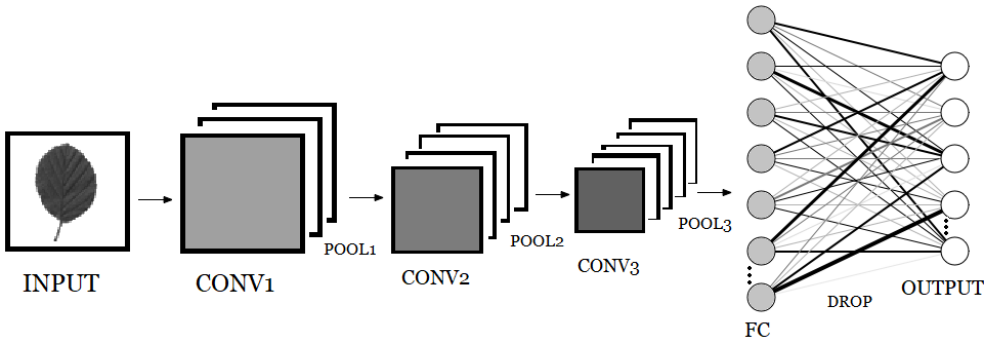

FIGURE 1. Proposed CNN architecture.

The INPUT layer refers to the grayscale input image. Every image has only one channel, so it is represented as a 2-dimensional matrix of pixel values. The matrix has a dimension of 64×64 and its elements are in the interval $[0, 1]$, where the value 0 indicates black and 1 indicates white.

The CONV1 layer has 8 filters, each being 5×5 square and uses *Leaky ReLU* as the activation function. The result of this layer is composed of 8 feature maps, each having the dimension 64×64.

The POOL1 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 8 feature maps, each having the dimension 32×32.

The CONV2 layer has 16 5×5 filters with *Leaky ReLU* activation. The result of this layer is composed of 16 32×32 feature maps.

The POOL2 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 16 16×16 feature maps.

The CONV3 layer has 32 5×5 filters with *Leaky ReLU* activation. The result of this layer is composed of 32 16×16 feature maps.

The POOL3 layer applies Max Pooling with a 2×2 window and a stride equal to 2. The output of this layer is represented by 32 8×8 feature maps.

The feature maps that resulted from the previous layer are flattened and concatenated, so they can serve as input for the FC layer.

The FC layer consists of 128 neurons, each having $8 \times 8 \times 32 = 2,048$ weights and 1 bias. Again, the *Leaky ReLU* activation function is used.

The DROP layer executes the dropout operation on the output of the FC layer, with 70% probability that an element will be kept.

The OUTPUT layer is actually a fully-connected layer, but with a different activation function: Softmax. It has one neuron for each possible class, and

they are connected to every neuron from the DROP layer, so they each have 128 weights and 1 bias.

5. Experimental Results

5.1. **Data sets.** Before evaluating the performance of network on the leaf datasets, some tests were performed on the benchmark dataset MNIST [11]. MNIST is composed of images with handwritten digits and the version used for training contains a total of 1,797 samples.

The **MEW2012** (Middle European Woods 2012) [13] leaf dataset consists of images of leaves collected from trees and shrubs native or frequently cultivated in the Czech Republic. The 2012 version contains 153 species, each having between 50 and 99 samples, with a total of 9,745 images. Only 20 of those species were used for experiments. The species were randomly selected, with a total of 1,240 images.

The **Swedish** [14] leaf dataset contains 15 species of trees from Sweden, with 75 samples for each of them, giving a total of 1,125 images.

5.2. **Image pre-processing.** The original images from the datasets had to be modified prior to being used by the Convolutional Neural Network. The reason is that the architecture of the CNN requires input images to have the same dimension and for each pixel to have only one corresponding value, restrictions which the samples from the datasets do not uphold. So, before the training process could begin, each leaf image was converted to grayscale and re-dimensioned to a 64×64 size. Because both datasets are quite small for the methodology employed, they were augmented. For each image, three more were generated by flipping the image, rotating it 5 degrees to the right and rotating it 5 degrees to the left.

5.3. **Training.** The CNN presented in this paper is trained using stochastic gradient descent because there is a certain level of redundancy in the leaf datasets used. The training set is composed of 80% of the images from the whole set, randomly selected. The rest of the dataset is used for testing purposes.

Throughout the training process, as well as afterwards, the performance of the network was measured through loss, accuracy and precision. Ideally, after each iteration, an evaluation of the network would reveal a smaller loss value, and a greater accuracy and precision. This, of course, is not always true. Sometimes, those values end up in some local optimum and the performance of the network has to get a bit worse before it can get better.

5.4. **Experiments.** The network architecture presented was trained using, in turn, each of the datasets mentioned. In the case of the leaf datasets, the experiments were conducted using the original datasets and also using the augmented version of each dataset.

5.5. **Results and discussion.** The accuracy, precision and loss obtained on the test data with the proposed CNN architecture are reported in Table 1.

Dataset	**Classes**	**Samples**	**Accuracy**	**Precision**	**Loss**
MNIST	10	1,797	97.33%	97.38%	0.0861
MEW2012	20	1,240	84.89%	86.13%	0.4831
MEW2012 + aug	20	4,960	**92.16%**	**92.30%**	0.2790
Swedish	15	1,125	**96.00%**	**96.32%**	0.1824
Swedish + aug	15	4,500	94.77%	94.85%	0.1703

TABLE 1. Results obtained on the test data for MNIST, MEW2012 and Swedish datasets.

With a simpler benchmark dataset such as MNIST, the network performed very well, which is always the first sign of a good architecture. With MEW2012, the image augmentation improved the performance dramatically, which means that the extra samples generated helped the model to generalize better. On the other hand, with Swedish, the best performance was obtained without augmentation, but the difference between the model trained using the original dataset and the one trained using the augmented dataset is not substantial.

A comparison can be made between the best results obtained with the proposed approach and the ones from the published papers that also employ CNNs in order to perform similar leaf recognition tasks. He et al. [6] performed their experiments on the ICL dataset, which was not publicly available at the time this article was being written. Jassman [7] performed the experiments on a personal dataset, which was not made public. Consequently, the comparison with these papers is made based on the number of classes in the datasets and the performance measurements used.

The CNN architecture presented in this paper achieved a precision of 92.30% on MEW2012, while He et al. [6] achieved 91.90% on ICL. Also, with the presented approach, an accuracy of 96.00% was obtained on Swedish, while Jassman [7] only obtained 56.66% accuracy on their private dataset. So, the proposed CNN architecture achieves higher precision for 20 species than He et al. [6] and higher accuracy for 15 species than Jassman [7]. This indicates that the proposed network is better fit for this task, but the comparison is not

conclusive due to the fact that the experiments couldn't be performed on the same datasets.

No clear comparison could be made with Atabay [1] because it is not understandable whether the performance measurement used is accuracy or mean average precision. Also, in their approach, the dataset is split in train (70%), validation (10%) and test (20%). But only the results on the validation set are presented, without providing the test results.

6. Conclusion

This paper presents a machine learning technique for leaf recognition. The chosen method is a Convolutional Neural Network due to the robustness and effectiveness of this kind of networks in image recognition tasks in general.

Experimental results with the MEW2012 [13] and Swedish [14] leaf datasets show that the proposed approach outperforms the other approaches that use CNNs for plant recognition in the literature.

During the development process, certain issues were encountered, such as overfitting, dying ReLU and numerical instabilities. But, by overcoming them, the designed model became much more reliable.

A more complex architecture might be able to recognize the species from the image without the leaf having to be placed on a white background. As the results obtained were promising, future work is under consideration in order to improve the current convolutional network implementation.

References

[1] H.E. Atabay. A convolutional neural network with a new architecture applied on leaf classification. *IIOAB Journal*, 7:326—-331, 2016.

[2] İ. Çuğu, E. Şener, Ç. Erciyes, B. Balcı, E. Akın, I. Önal, and A. O. Akyüz. Treelogy: A novel tree classifier utilizing deep and hand-crafted representations. *arXiv preprint arXiv:1701.08291*, 2017.

[3] P. Dahal. Classification and loss evaluation - softmax and cross entropy loss. https://deepnotes.io/softmax-crossentropy.

[4] C. Y. Gwo and C. H. Wei. Plant identification through images: Using feature extraction of key points on leaf contours. *Applications in plant sciences*, 1(11), 2013.

[5] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[6] X. He, G. Wang, X. P. Zhang, L. Shang, and Z. K. Huang. Leaf classification utilizing a convolutional neural network with a structure of single connected layer. In *International Conference on Intelligent Computing*, pages 332–340, 2016.

[7] T. J. Jassman. Mobile leaf classification application utilizing a convolutional neural network. Master's thesis, Appalachian State University, 2015.

[8] R. Kapur. Rohan #4: The vanishing gradient problem. https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b.

[9] A. Karpathy. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[10] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and J.V. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *Computer vision–ECCV 2012*, pages 502–516. Springer, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] M. E. Nilsback and A. Zisserman. *An automatic visual flora: segmentation and classification of flower images*. PhD thesis, Oxford University Oxford, 2009.

[13] P. Novotnỳ and T. Suk. Leaf recognition of woody species in central europe. *biosystems engineering*, 115(4):444–452, 2013.

[14] O. Söderkvist. Computer vision classification of leaves from swedish trees. Master's thesis, Linkoping University, 2001.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[16] M. Sulc and J. Matas. Texture-based leaf identification. In *European Conference on Computer Vision*, pages 185–200. Springer, 2014.

[17] A. Wendel, S. Sternig, and M. Godec. Automated identification of tree species from images of the bark, leaves and needles. In *16th Computer Vision Winter Workshop*, page 67. Citeseer, 2011.

[18] S. G. Wu, F. S. Bao, E. Y. Xu, Y. X. Wang, Y. F. Chang, and Q. L. Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 11–16. IEEE, 2007.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 Kogălniceanu, Cluj-Napoca, 400084, Romania

Email address: olic1770@scs.ubbcluj.ro