Leaf Classification Using Convolutional Neural Networks and Pre-Extracted Features

Omid Abdollahi Aghdam
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
abdollahi15@itu.edu.tr

***Abstract*—It has been always challenging for experts to classify plant using leaf images and when it is done manually is prone to duplicate identifications. In this paper we use a dataset of 99 plant species which contains 16 leaf images per class and introduce a method to leverage pre-extracted features plus features extracted by convolutional neural networks (CNN) of binary images of plant species to automate the classification task and minimize the inaccuracy. The experimental results show that combining both aforementioned features leads to higher accuracy and lower multi-class logarithmic loss.**

Keywords—leaf classification; convolutional neural networks; deep learning

I. INTRODUCTION

Presence of plants is vital as they are the source of nearly half of the oxygen in the earth and food resources for other species and human beings . Thus, studying their population is a must. It has been always a challenging task for human to classify the plant species. Therefore, automatic leaf classification techniques will help us to track and preserve plant species population, to study plant-based medical research, and will ease the crop and food supply management [1]. Taking into account the aforementioned statements, kaggle.com[1] has conducted a competition for leaf classification. The goal of the competition is to use binary leaf images and pre-extracted features, to classify the 99 species of plants. The CNN is the state-of-the-art technique in the field of computer vision and it has been the wining method in the ImageNet[2] challenges for the last few years [2, 3, 4, 5]. In this work, the CNN is used to learn unsupervised feature representation of images which then is fed to fully connected layer alongside with pre-extracted features. The experimental results show, having combined both feature representations leads to lower multi-class logarithmic loss in classification of leaves compared to methods which use only pre-extracted features or features extracted by CNN.

II. RELATED WORKS

There are many works in the literature for classifying the plant species from which we briefly mention three works which are more similar to our work. Two works used CNN to extract unsupervised features of leaf images for classification of plants and one work in which authors used pre-extracted features. In the work by Zhiyu Liu et al. [6], they used CNN for extracting the features of the plant leaves, and then used those features to classify leaves with SVM. The highest accuracy reported in their work is 93%. In another work by Sue Han Lee et al. [7], Convolutional Neural Network is used to extract features from images, and the Multi-Layer Perceptron classifier is used to classify the leaves. The highest accuracy reported in their work is 99.5%. Finally, Charles Mallah et al. [8] used density estimation from a K-NN classifier with probabilistic integration of shape, texture, and margin features. They achieved 96% accuracy in the classification task. As it is shown in the flowing sections, combining both features will be resulted in higher accuracy.

III. DATASET

In the dataset provided on kaggle.com for this competition there are 1584 images of leaves and two CSV[3] files, one for training set and one for test set [9]. Each of leaf specimens has 16 samples and there are 99 species of leaves. The dataset is divided into train set, 990 images, and test set, 594 images, there is a sample of leaf images shown in the Fig. 1. Moreover, three sets of features (margin, shape, texture) are provided per image in the CSV files which are represented by a 64-attribute vector. In other words, there are a 192-attribute vector per leaf sample. There is an anonymous id, unique to an image and the classes of leaf specimens are not provided in the test set. Thus, we have to submit the result to evaluate our model on the test set on which we are provided by a score of multi-class logarithmic loss, and we are allowed for 5 submission per day.

1 Kaggle.com is a platform on which which companies and researchers post their data and statisticians and data miners from all over the world compete to produce the best models.

2 The *ImageNet* project is a large visual database designed for use in visual object recognition software research.

3 In computing, a comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

Fig. 1. A sample of leaf images in the dataset.

IV. Brief Introduction To CNN

In this section, the Convolutional Neural Network architecture will be explained. The Convolutional Neural Networks take as input pixel values of image and convert them into class scores by passing them through layers which are described in details in the following subsections. There is a schematic diagram of the layers of CNN in Fig. 2.

A. Input layer:

The Input Layer receives the raw pixel values of the images. The images used in this work are scaled to images of width 128, height 128, and with a binary color channel, zero for black and 255 for white.

B. Convolutional layer:

The CONV layer's parameters consist of a set of learn-able filters. For example, in this work filters of the first layer of the ConvNet has size 3x3x1 (i.e. 3 pixels width and height, and 1 because images are binary and have depth 1, the color channels). Each filter is convolved across the width and height of the input volume and compute dot products between the weights of the filter, which are initialized randomly, and the input at any position. As a result, a 2-dimensional activation map that gives the responses of that filter at every spatial position is learned by CNN [10].

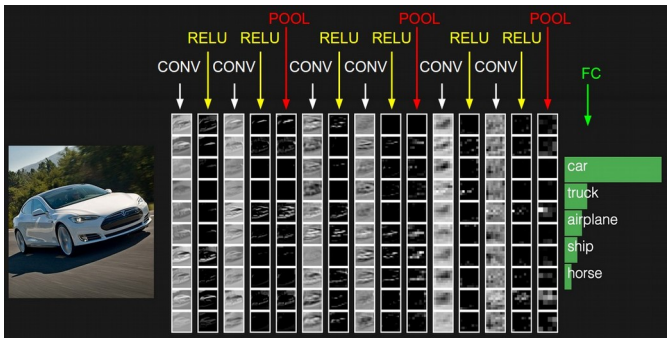

Fig. 2. CNN takes an image as input and convert the pixel values into class by passing them into multiple layers [10].

Intuitively, the network will learn filters that are activated when they see some type of visual feature such as an edge of some orientation. Usually, there are a set of filters in each CONV layer (e.g. 8 filters), and each of them will produce a separate 2-dimensional activation map. These activation maps will be stacked along the depth dimension and produce the output volume.

The spatial size of the output volume can be computed using input volume size (W), the filter size of the CONV Layer (F), and the stride size of them (S). The stride size is the number of pixels that CONV Layer filters slide on the image.

- Input dimension: W1 H1 D1
  - W1: Width of input
  - H1: Height of input
  - D1: Depth of input
- Output dimension: W2 H2 D2
  - F : Filter size
  - P : The amount of zero padding which is used to preserve the spatial size of the input
  - W2 = (W1 - F + 2P)/S + 1
  - H2 = (H1 - F + 2P)/S + 1
  - D2 = K (K is the number of filters in the previous CONV layer)
- With parameter sharing, it introduces F.F.D1 weights per filter, for a total of (F.F.D1).K weights and K biases [10].

A visualization of input image and 2-dimensional feature maps which are learned by the CNN layers in this paper is shown in Fig. 4, 5, 6.

C. Activation layer:

In this work, rectified linear unit (ReLU) is used as the activation layer. This layer uses the following function to eliminate the values less than zero.

$$f(x) = \mathrm{Max}(0,x)$$

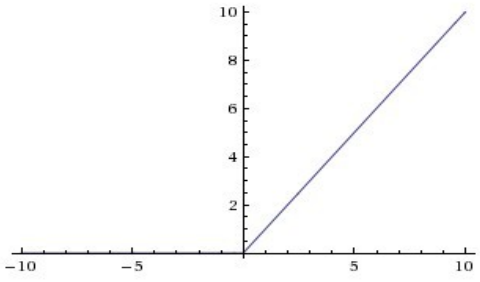

Fig. 3. ReLu Function

Fig. 4. An input image

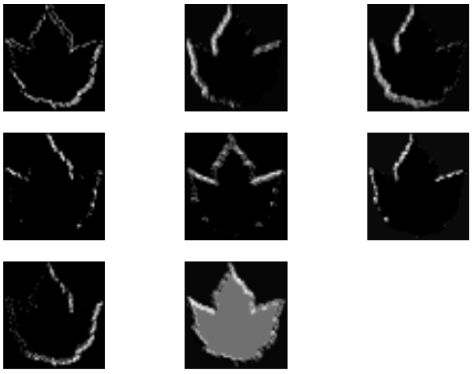

Fig. 5. Visualization of filters of the first convolutional layer which has 8 filters.

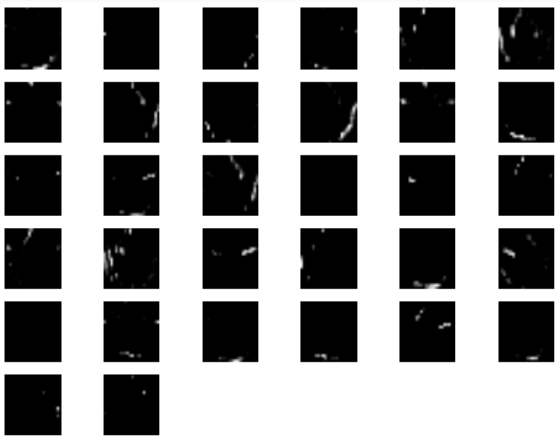

Fig. 6. Visualization of filters of the second convolutional layer which has 32 filters.

D. *Pooling layer:*

The Pooling Layer is inserted between CONV layers and is used to reduce the spatial size of the representation in order to decrease the number of parameters and in turn computation cost in the network. There are two common Pooling Layer, namely, Max Pooling and Average Pooling. In this case, Max pooling with the filter size of 2 is applied. The pooling layer takes as input samples of size 2*2 pixel and output the maximum value of the samples. In other word, Pooling Layer down-sample the volume spatially [10].

The spatial size of the output volume of Pooling layer also can be computed using input volume size (W), the spatial extent of Pooling Layer (F), and the stride size of them (S).

- Input dimension: W1 H1 D1
  - W1: Width of input
  - H1: Height of input
  - D1: Depth of input
- Output dimension: W2 H2 D2
  - F : Filter size
  - W2 = (W1 - F)/S + 1
  - H2 = (H1 - F)/S + 1
  - D2 = D1

An example of max pooling layer is shown in Fig. 7.

E. *Fully connected layer:*

The Convolutional Neural Network Layers are followed by one or multiple Fully Connected layer, commonly, a Multi-Layer perceptron. The Neurons in these layers have full connections to all the activations in the previous layer. In the final Fully Connected layer, the softmax[4] function is often used which outputs the probability distribution of classes for the input images.

F. *Merging features extracted by CNN with pre-extracted features:*

In this work, before passing the output of CNN to fully connected layers, learned features are flattened and merged with 192 pre-extracted attributes per image. Afterward, the merged values are passed through Fully Connected Layer. For example, if the input image size is 128*128*1 and we use two layers Convolutional Neural Network with filter size of 3, the stride size of 2, and the pooling size of 2, the CONV Layers will produce 31*31*32 3-dimensional map per image which

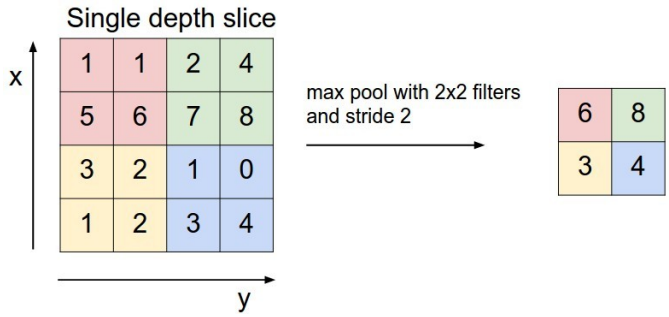

Fig. 7. Max pooling layer takes 2*2 matrix of sample downsampling them with maximum value.

4 cs231n.github.io/linear-classify/#softmax

will be flattened to a vector of size 30752 and concatenated with 192 pre-extracted features to shape a attribute vector of size 30944 for representation of each image.

G. *Evaluation method:*

The multi-class logarithmic loss (categorical cross entropy) is used to evaluate the results [11].

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

Where N is the number of images in the test set, M is the number of species labels, log is the normal logarithm, yij is 1 if observation i is in class j and 0 otherwise, and pij is the predicted probability that observation i belongs to class j. For minimizing the loss function RMSprop[5] optimizer is used to minimize the error using back-propagation[6].

V. Our Approach

In this work, multiple architecture of CNN have been tried to classify the leaves, the architecture and results for the top five model is summarized in TABLE I. As there is not many images for each class we used data augmentation method to increase the number of samples artificially. Data augmentation reduces over-fitting on image data. The architecture we used in this work is a two or three Conv layer each followed by a ReLU and Pooling layer which extract the unsupervised feature representation of the input leaf images. Afterward, we flattened the newly extracted features and merged them with 192 pre-extracted three set of features, shape, texture, margin. Later, we passed the results to fully connected layer in which we used softmax function to output the probability distribution of classes for the input leaf images. In the fully connected layer, dropout technique is used to reduce over-fitting. The dropout is a simple way of reducing over-fitting in artificial neural networks in which user specify a percentage for dropping some of connections randomly. In this work, 50% dropout is used. The results of architecture is evaluated using multi-class logarithmic loss function and RMSprop optimizer is used to update the parameters in each iteration. Furthermore, 10-fold cross-validation is used for evaluating the model. One hundred epoch is used in this work. Keras[7] library with Tensorflow[8] backend in python programing language is used for constructing the architecture. Please follow the instruction on [13] to run the code.

5 http://cs231n.github.io/neural-networks-3/#ada

6 http://cs231n.github.io/optimization-2/#intuitive

7 Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research. https://keras.io/*

8 An open-source software library for Machine Intelligence. https://www.tensorflow.org/

VI. Results

As it stated, the class value of test set is not released on kaggle.com, and kaggle compare the results of each submission based on multi-class logarithmic loss after submitting the predictions for test set. There are 32 submission of the results of this work on kaggle.com with the best Multi-class logarithmic loss of 0.00709, ranking 29 amongst 1143 teams. An screen-shot of the public leaderboard is shown in Fig. 8 (username: UmuDev) [12]. The architecture and the results of top 5 submission are shown in TABLE I. Moreover, Model accuracy per epoch for training and validation set, and Model loss per epoch for training and validation set for the best model is shown in Fig. 9, and Fig. 10 respectively.

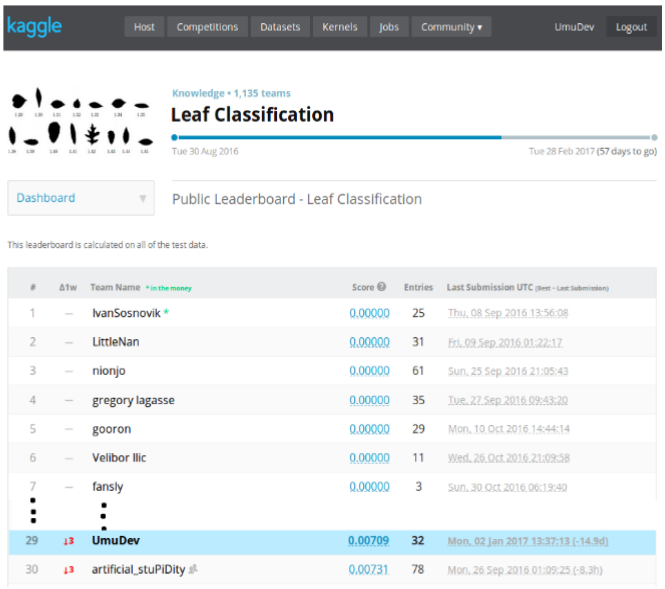

Fig. 8. Public Leaderboard for leaf classification on kaggle.com

TABLE I. The Architectures and Results for the Top Five Model.

Architecture	Validation Accuracy	Validation loss	Test loss
Input image size: 128*128*1 Conv1: 8*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv2: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 FC1: 512 Dropout: 0.5 Out: 99 softmax	1	0.00001	0.00709

Input image size: 128*128*1 Conv1: 16*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv2: 16*3*3 ReLU - Pool: 2*2 - Stride: 2*2 FC1: 1024 Dropout: 0.5 Out: 99 softmax	1	0.00262	0.03041
Input image size: 128*128*1 Conv1: 16*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv2: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 FC1: 1024 Dropout: 0.5 FC2: 512 Dropout: 0.5 Out: 99 softmax	1	0.00256	0.03267
Input image size: 128*128*1 Conv1: 16*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv2: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv3:: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 FC1: 200 Dropout: 0.5 FC2: 100 Dropout: 0.5 Out: 99 softmax	1	0.00019	0.03454
Input image size: 128*128*1 Conv1: 16*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv2: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 Conv3:: 32*3*3 ReLU - Pool: 2*2 - Stride: 2*2 FC1: 200 FC2: 100 Out: 99 softmax	1	0.0067	0.03940

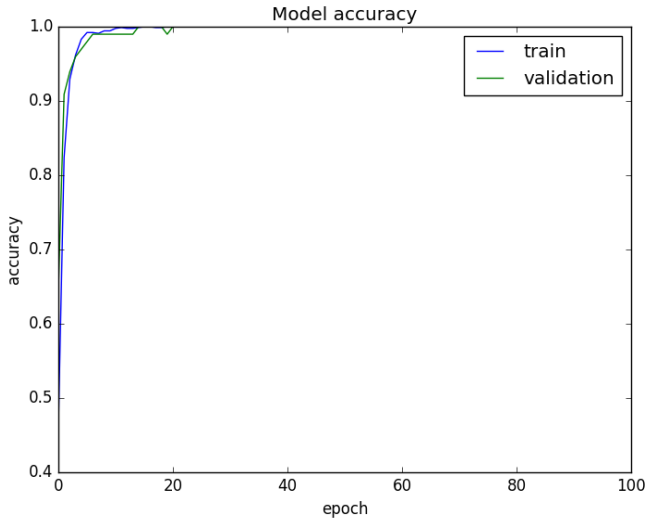

Fig. 9. Training and Validation accuracy based on the number of epoch

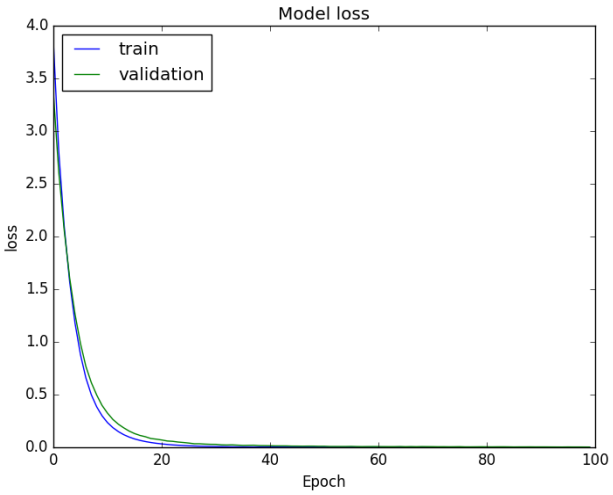

Fig. 10. Training and Validation loss based on the number of epoch

VII. Concolution

As the experimental results show, merging the pre-extracted features with the features extracted using CNN can lead to higher accuracy and lower loss in the validation set. Furthermore, it means a lower loss in the test set which can be seen after uploading the prediction results of the test set on kaggle.com. In this work, 10-fold cross-validation have been used to evaluate the models. The evaluation method used in this work is categorical cross entropy (Multi class logarithmic loss) and the model is optimized using RMSprop. Moreover, it has been observed that deeper CNNs and Fully Connected layers increase the loss. It might be the result of overfitting which can be investigated in the future works. Adding the

Dropout layers to fully connected layers can prevent overfitting. In this work 0.5 Dropout layer is used in which half of the neurons are dropped randomly during training. Using this techniques, we managed to reduce the loss of test set to 0.00709. The architecture resulted to it is presented on the first row of TABLE I.

REFERENCES

[1] Leaf Classification. (n.d.). Retrieved January 08, 2017, from https://www.kaggle.com/c/leaf-classification

[2] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1106–1114, 2012.

[3] K. Simonyan, A. Zisserman. Very deep convolutional networks dor large-scale image recognition. In International conference on learning representations 2015. ICLR 2015.

[4] C. Szegedy, et. al. Going Deeper with Convolutions. In Computer Vision and Pattern Recognition, 2015. CVPR 2015. IEEE Conference on, 2015

[5] K. He, X. Zhang, Sh. Ren, J. Sun. Deep Residual Learning for Image Recognition. In Computer Vision and Pattern Recognition, 2016. CVPR 2016. IEEE Conference on, 2016

[6] Liu, Z., Zhu, L., Zhang, X., Zhou, X., Shang, L., Huang, Z., & Gan, Y. (2015). Hybrid Deep Learning for Plant Leaves Classification. *Intelligent Computing Theories and Methodologies Lecture Notes in Computer Science*, 115-123.

[7] Lee, S. H., Chan, C. S., Wilkin, P., & Remagnino, P. (2015, 09). Deep-plant: Plant identification with convolutional neural networks. *2015 IEEE International Conference on Image Processing (ICIP)*.

[8] Mallah, Charles, James Cope, and James Orwell. "Plant Leaf Classification Using Probabilistic Integration of Shape, Texture and Margin Features." *Computer Graphics and Imaging / 798: Signal Processing, Pattern Recognition and Applications* (2013).

[9] Dataset of Leaf Classification. (n.d.). Retrieved January 08, 2017, from https://www.kaggle.com/c/leaf-classification/data

[10] CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved January 08, 2017, from http://cs231n.github.io/convolutional-networks/

[11] Evaluation method ofLeaf Classification. (n.d.). Retrieved January 08, 2017, from https://www.kaggle.com/c/leaf-classification/details/evaluation

[12] Public Leaderboard of Leaf Classification. (n.d.). Retrieved January 08, 2017, from https://www.kaggle.com/c/leaf-classification/leaderboard

[13] Reademe file on my google docs account: https://goo.gl/yY2fRa