EEE 511 - Assignment 5
Plant Leaf Recognition Using Self Organizing Maps

Xiao Huang, Suhas Lohit, Rakshit Raghavan
1206424709, 1206451918, 1207751060

***Abstract*—In this assignment we demonstrate the use of Self Organizing Map (SOM) as a technique for plant leaf recognition. Given a set of leaves from different plants, we train the SOM using different images of the same leaf. Once the network is trained, it can be used to identify plants when a particular leaf picture/image is the input. Such algorithms can be useful for travelers, botanists and other plant enthusiasts. A number of geometrical aspects and other features of each plant leaf are extracted using image processing techniques. Once these features are available, they are used as training data for the SOM and further analysis. We use the publicly available *Folio* dataset containing leaf images belonging to 32 different plants. We compare the accuracy obtained by SOM with a k-Nearest Neighbor (k-NN) classifier and a Support Vector Machine (SVM).**

***Index Terms*—Plant leaf recognition, shape features, self organizing map.**

I. Introduction

HUMAN beings depend on plants for food and air. Plants play a vital role in the survival of mankind and hence, their study has gained wide respect and following over the years. Many botanists can identify plants by just glancing at their leaf. However, the exact number of plant species is still under scrutiny and experts have predicted that about 400,000 of them are in existence. Thus, it becomes necessary to study, classify and create a database about them so that we may know when new species are discovered and when a particular plant is getting extinct. This is where machine learning techniques comes in the picture and can provide massive support in the background for what seems to be a simple leaf recognition technique.

With so many plants all over the world, we can employ simple methods such as root recognition, stem or trunk observations etc. But these techniques are obsolete and less accurate. Botanists and chemists may extract plant genes or molecules and employ their own plant recognition systems. These again would be less accurate, time consuming and expensive. A cheaper and better alternative is to have a database of plant images/photographs captured using a simple digital camera. Once, these are available, we can employ various image processing and feature extraction techniques. These techniques are advantageous since - they are fast, efficient, robust, flexible and more accurate.

Once the database is available, the next task is to extract features which the machine/ neural network can recognize. A number of plant feature extraction techniques have been discussed before. Skeleton segmentation of plants using Gaussian and wavelet transformation technique has been mentioned by Gu et al. in [1] Another pattern classification method prosposed by Wang et al. in [2] using a hypersphere that moves along a median center or Moving Median Center (MMC). Our basic feature extraction methods coincide with those as mentioned by Munisami et al. in [3] where they have extracted basic geometrical features of an image leaf such as perimeter, area, x and y distance maps, centroid ratios. These features are simple and easy to calculate and feed to the machine learning model since there are more than 600 image leaves in the database and calculating these features are both time efficient and accurate.The *Folio* dataset is available on the UCI repository [4] and consists of 32 plants with about 20 images of each plant leaf.

As described in the paper, *Munisami et al.* have employed a k-NN classifier for their plant recognition algorithm. In this assignment, we have followed all steps as mentioned by them for feature extraction. However, we have tried to generate results using Self Organizing Map (SOM) method instead of the k-NN classifier. SOM is comparatively useful since the analysis of high dimensional data becomes easier in a lower dimensional space [5]. The arrangement of these maps, preserve topological ordering and provides useful insight as images with similar features are mapped together and we can analyze better. In particular, we have used the *Folio* dataset and split it into training and testing datasets and use it for obtaining further results.

This report is organized as follows. In Section II, we describe various image preprocessing techniques that we employ on each image before extracting the features. Section III explores the main features that have been obtained from each processed leaf which will be used ahead for training the SOM network. In Section IV we describe the SOM algorithm and its visualization. Further, in Section V we explain how we the training and test data are split. Finally, in Section VI we have compared the results that we obtained with our implementation of [3].

II. Pre-Processing the Folio Dataset

The Folio dataset contains images of leaves. Some examples of the images are shown in Figure 2. We follow the same preprocessing steps as in [3]. As shown in figure 1 the pre-processing step for each image consists of a number of tasks. Each original image obtained from the repository is of very high resolution with dimensions of 4128 X 2322 pixels. Operating on so many images with such high dimensions is unnecessary and time consuming. Hence, the foremost step

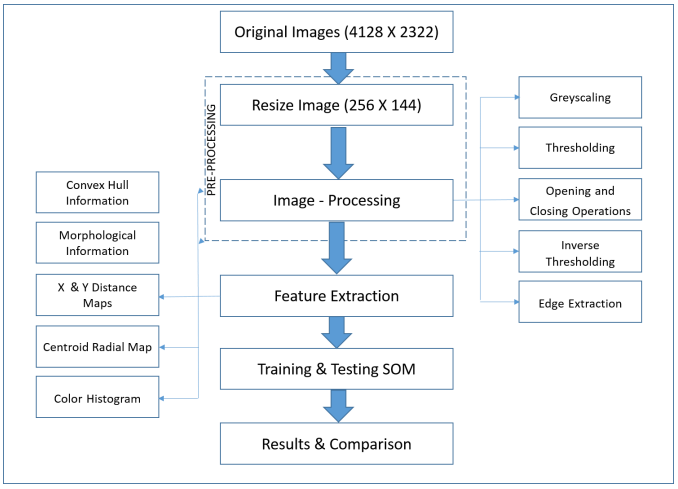

Fig. 1. Overview of our plant leaf recognition algorithm

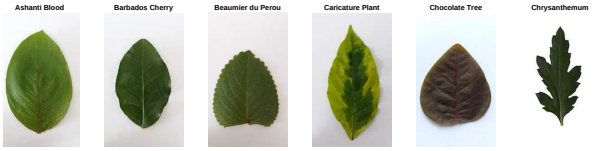

Fig. 2. Sample images from the Folio dataset

is to resize the image, maintaining the aspect ratio to 1.77. Hence, we resize each image to 256 X 144 pixels. Once the resized image is obtained, each of them go through a number of image processing steps as described below.

A. Grayscaling

Since the preprocessing is done to extract geometric features, the information in the images is of no importance to us anymore. All we need is to extract information from the contours of each leaf image. Hence, we convert the images into grayscale represent each pixel with regards to its intensity. The image is now stored in the new dimensions but the RGB image is converted to an image in **'grayscale'** format.

B. Thresholding

Once the grayscale image is obtained, we perform a thresholding operation using Otsu's method. This method sets a threshold and forms clusters of images. Pixels that are above the threshold are set to **'1'** while those below the threshold are set to **'0'**

C. Opening and Closing Operations

Once the image is converted into a binary image, holes are left behind and these need to be removed for further processing. Hence, a morphological image processing technique is applied in order to removed these unwanted features/imperfections from the image as described below

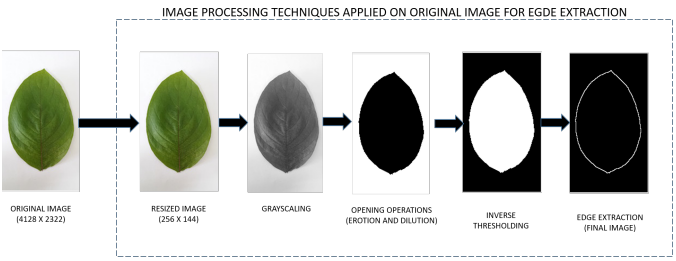

Fig. 3. Example of an *'Ashanti Blood'* leaf where image processing is applied to obtain final image for feature extraction

1) Erosion: The binary image is *eroded* using a diamond structure. The elements of the diamond structure is place on the foreground of the binary image and only those elements common to both are extracted. This process is repeated until the diamond moves across all pixel coordinates of the image.

2) Dilation: The eroded image is dilated as explained above to obtain a new binary image that is cleaned up and free of holes and other added noises.

D. Inverse Thresholding

A simple inverse operation is performed where the pixel values are reversed from 0 to 1 and vice versa in order to obtain a black background.

E. Edge Extraction

The edges need to be extracted for further processing. Hence, we use *Robert's* Algorithm. Robert's algorithm approximates derivative of an image at each point and returns those points as edges whose derivative is maximum. Note that this feature is different from [3] where they use Suzuki's algorithm.

III. Feature Extraction

We extract very similar features as in [3]. Once the image pre-processing is complete and the boundry points of each image is extracted, we obtain the following features from each image.

A. Convex Hull Information

Convex Hull information of an image gives the smallest set of points within which the image is contained. We use this information to calculate the perimeter and area of the convex hull around the processed image.

B. Morphological Information

Features such as length, width, perimeter and area of the processed image leaf are calculated and stored. The minimum and maximum **x and y** coordinates of the image aid in these calculations.

C. X and Y Distance Maps

We have divided the image into ten horizontal and vertical segments. A boundary box is formed around the image. The distance of each horizontal segment of the leaf to the nearest image vertical boundary is stored as an x-map. Similarly the distance of each vertical segment of the leaf to the nearest image horizontal boundary is stored as an y-map.

D. Centroid Radial Maps

We divide each leaf image and the boundary box surrounding it into 16 equal segments. The distance of the centroid to these segments is computed and stored for further computations.

E. Color Histogram

The cropped version of the image, containing only the leaf, is used for this purpose. It needs to be ensured that no background (or white part of the leaf) is taken into consideration. A color histogram of this reference part of the image is computed and stored.

Once above features are calculated. we formulate the features dataset that is to be used ahead to train the SOM. The dataset is a 637 X 58 matrix for the 637 images and 58 features. The table showing these dataset features and their column distributions is shown below

IV. Self-Organizing Map

A self-organizing map (SOM) is an unsupervised, dimensionality reduction/clustering technique that organizes the input space in a discrete representation according to their features. The input vectors must have common features/patterns that are recognized by this artificial neural network and it forms a topographical representation of these vectors in a lower dimensional space.

Unlike other techniques that update a single neuron, SOM works on a neighborhood update principle. That is, updates are applied not only to the closest or **winner** neuron but also to the neighborhood of this neuron.In this way, a number of neurons move towards the same input.[6] Below equation shows the mathematical principle underlying this technique

$$\Delta m_l = \eta e_{l,i}(x^t - m_l) \tag{1}$$

where, i, is the best matching unit (BMU), η is the learning rate and $e_{l,i}$ is the neighborhood function.

The use of neighborhood function has a couple of advantages. Firstly, there is an elimination of dead units as more than one neuron is updated and they get to become a winner neuron eventually. Secondly, random initialization problem has negligible effects on the system since neurons move together towards an input space and neighboring neurons represent similar inputs [7].

In principle, the SOM tries to topographically arrange its inputs in 1 or 2 dimensions with the advantage being of a highly orderly fashioned arrangement. Neurons formed as a result of this need to be connected to each and every input in the higher dimensional space [8]. In order to get a proper neuron lattice formation, we need sufficient training data for the SOM to learn and organize itself effectively.

V. Dataset for training and testing

As mentioned earlier, the dataset used for training and testing in this assignment is a publicly available dataset called Folio [4]. It contains 637 leaf images belonging to 32 different plants with about 20 images per plant. Thus, the problem is a 32 class classification problem. For each image (after preprocessing), we calculate a 58-dimensional vector as described in Section III. We use 17 images from each class as part of the training set and the remaining as the testing set. In the end, we have a training data matrix of size 544 x 58 and a testing data matrix of size 93 x 58 with the rows as data vectors and the columns containing features as described in Table I.

VI. Experimental Results

In this section, we describe in detail, the SOM used for training including the setting of various parameters. We compare the recognition accuracy obtained using the SOM against two other algorithms - k-NN, as used in [3] as well as a linear SVM. Note that we are comparing the performance of the SOM, which is an unsupervised learning algorithm against two supervised algorithms.

A. SOM description

We use the SOM Toolbox available online [6] We use an SOM with a hexagonal lattice structure with a grid of size 22 x 10 units. The neighborhood function used is Gaussian. The training algorithm is run in batch mode. For the training phase, each of the 544 training data vectors is presented to the SOM and the weights are updated until convergence or until the maximum number of iterations has been reached. We use the `som_make` function for this purpose. Then, using the `som_autolabel` function, we label the output nodes, using a voting scheme. In this scheme, we count the number of data vectors of each class that have a given node as the best matching unit (BMU) and the class that has most number of vectors mapped to this node is used to determine the label of the node. Thus, during training, we do not make use of the labels provided to us and hence, this is an example of an unsupervised learning algorithm. For testing, we determine the BMU's (using `som_bmus` function) for each of the test vectors and the compare the labels of the BMU node (the SOM prediction) of each vector with the ground truth label. This is used to determine the recognition accuracy of the SOM.

B. Baselines for comparison

We compare the results of algorithm with our implementation of [3] – k-NN algorithm as well as a linear SVM. Both of these are supervised algorithms that make use of the ground truth labels. The training and testing set are the same as before. For the k-NN, we set the value of $k = 3$. That is, the labels of 10 closest neighbors are used to predict the label of the a given test vector. For the SVM classification, we use the `multisvm` function in Matlab that uses the `svmtrain` function with default parameters to train 32 one vs all classifiers, one for each class. The predictions for the test set is obtained by using the outputs of all the 32 SVMs and the label of a given test vector is equal to the index of the SVM that provides a positive output.

Parameter	Plant Label	Aspect Ratio	White Area Ratio	Perimeter to Area	Perimeter to Hull	Hull Area Ratio	X-Distance Map	Y-Distance Map	Centroid Radial Distance	Color Histogram
Column(s)	1	2	3	4	5	6	7 : 16	17 : 26	27 : 42	43 : 58
Description	Labels assigned to leaves from 1 through 32	Width of Leaf/Length of Leaf	Area of Leaf / (Length * Width)	Perimeter of Leaf / Area of Leaf	Perimeter of Hull / Perimeter of Leaf	Area of Leaf / Area of Hull	Distance of each Horizonal Segment / Length of Leaf	Distance of each Vertical Segment / Width of Leaf	Distance from Centroid to each leaf Segment/Distance of Centroid to each boundary Segment	Color Histogram of cropped image

TABLE I
FEATURE VECTOR ELEMENTS AND THEIR DESCRIPTION.

C. Recognition accuracy on the Folio test set

Using all the 58 features for training and testing, the accuracy obtained by the SOM was much poorer than those of k-NN and SVM. It was then observed feature selection gave a remarkable boost in accuracy. The images in the dataset suggested that the Y-distance maps may not be encoding enough discriminating features. By removing the Y-distance map features, i.e., by using features 2 through 16 and 27 through 58, we obtained an improved accuracy of 82.8%. Using the feature set (excluding the Y-distance map), the accuracies using k-NN and SVM were 84.95% and 82.8% respectively. The paper [3], whose pipeline we have followed for feature extraction, uses k-NN and the authors report an accuracy of 83.5% in the first stage (all features except color histogram) which improves to 87.2% in the second stage (see [3] for more details).

Note that the linear SVM yields poorer results than the much simpler k-NN algorithm. This is an indication that the desired decision boundaries in the feature space are highly non-linear. SOM yields a poorer but a very competitive accuracy, in spite of being an unsupervised algorithm. This shows that the features extracted from the images combined with the SOM training algorithm produces clusters that are discriminative to a high degree. The SOM performance therefore can be increased further by extracting more discriminative features. The downside of this is that it requires more domain knowledge.

D. Visualization of SOM results

We show two kinds of visualizations for the trained SOM using all features except the Y-distance map vector. We use the visualization functions provided in the SOM toolbox [6]. This helps in the analysis of results. Figure 4 shows the plots of the code vectors for each of the SOM units or nodes. It can be seen that neighboring units have similar code vectors. This indicates that neighboring units tend to be activated by similar input vectors, thus preserving topological ordering.

Figure 5 shows the U matrix obtained. The U matrix indicates Euclidean distances between code vector of neighboring nodes. The colorbar shows that the blue elements indicate the corresponding pair of nodes have very close code vectors. The U matrix can be used as a rough 2D visualization of the high dimensional data as well as seeing the clusters present in that feature space.

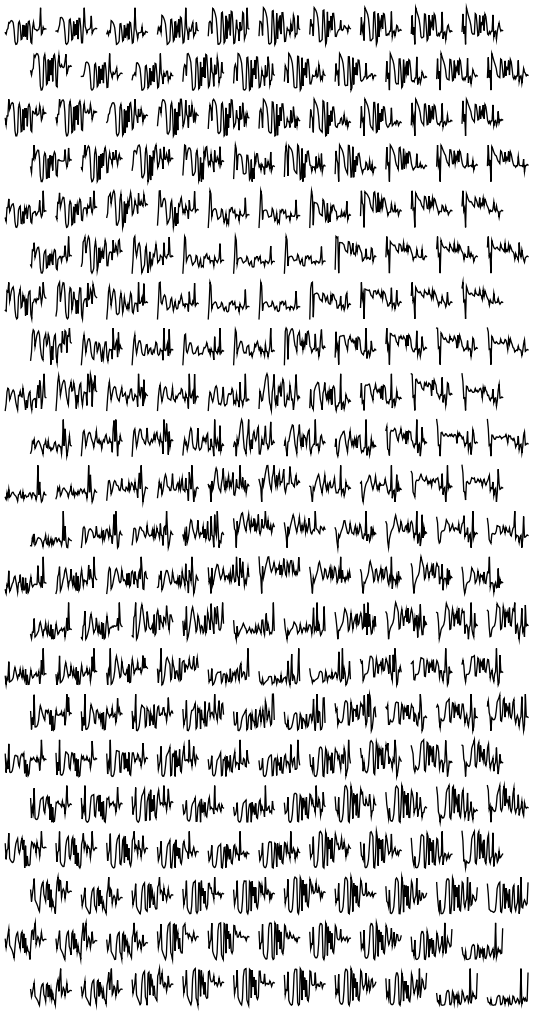

Fig. 4. Plots of code vectors for each of the SOM units. It can be observed that neighboring units tend to have similar code vectors. Zoom in for better readability.

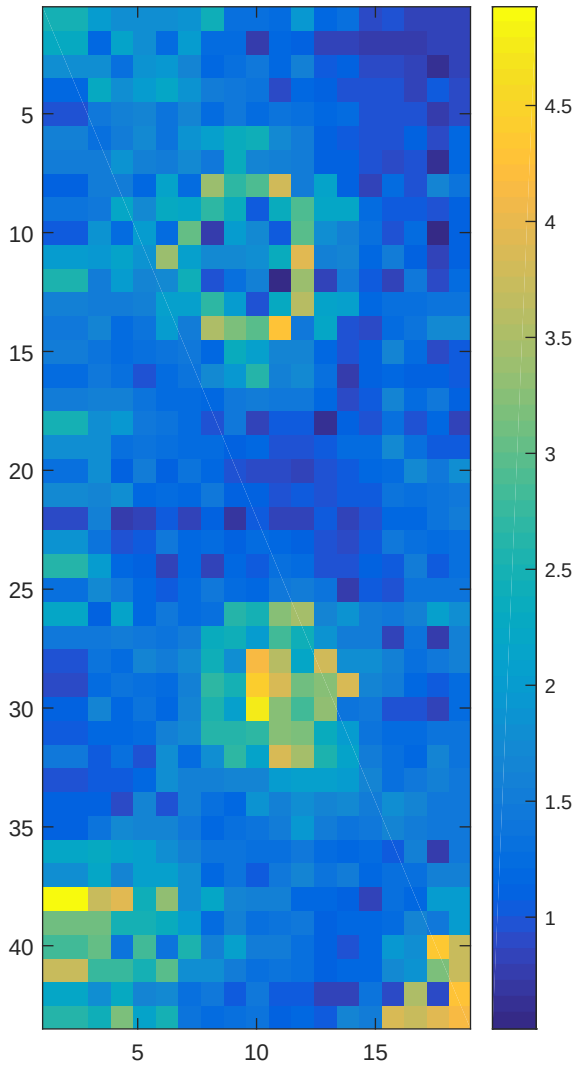

Fig. 5. U matrix of the trained SOM

VII. Running the code

All the codes were written in Matlab and are attached in the submission. The submission also includes the resized version of the dataset used. Also, the extracted features for all the images in the dataset are stored in `features.mat`. In order to compute the recognition accuracy on the dataset, it is sufficient to run the file **`som_leaf.m`**. This file splits the dataset into training and testing sets, trains the SOM, SVM and k-NN classifiers and reports the accuracies for each of the algorithms. In order to run the entire pipeline including feature extraction, run the `main_folio_features.m` file.

VIII. Conclusion

In this assignment, we trained a self organizing map, an unsupervised learning algorithm, for the purpose of plant leaf recognition. We compared our results to other supervised learning algorithms – k-NN and SVM. We show the SOM can provide very competitive results even though it is agnostic to the class labels during training. The clusters produced by the SOM are sufficient to provide a high recognition accuracy, even for a difficult dataset with large number of classes.

References

[1] X. Gu, J.-X. Du, and X.-F. Wang, Leaf recognition based on the combination of wavelet transform and gaussian interpolation, in Proceedings of International Conference on Intelligent Computing 2005, ser. LNCS 3644. Springer, 2005.
[2] X.-F. Wang, J.-X. Du, and G.-J. Zhang, Recognition of leaf images based on shape features using a hypersphere classifier, in Proceedings of International Conference on Intelligent Computing 2005, ser. LNCS 3644. Springer, 2005.
[3] Munisami, T., Ramsurn, M., Kishnah, S. and Pudaruth, S., 2015. Plant leaf recognition using shape features and colour histogram with k-nearest neighbour classifiers. Procedia Computer Science (Elsevier) Journal. 58, pp. 740-747.
[4] *Folio Data Set*, UCI Machine Learning Repository Link.
[5] *Self-organising Map*, Wikipedia Link.
[6] *SOM Toolbox*, Link.
[7] Ethem Alpaydin, *Introduction to Machine Learning*, Vol. 2. The MIT Press, Cambridge, 2010.
[8] Simon Haykin, *Neural Network and Learning Machines*, Vol. 3. Upper Saddle River, Pearson Education, 2009.