WARSAW UNIVERSITY OF TECHNOLOGY

Image Processing and Computer Vision

Faculty of Mathematics and Information Science

Tree Species Recognition

- Documentation -

By **Alaa Abboushi**

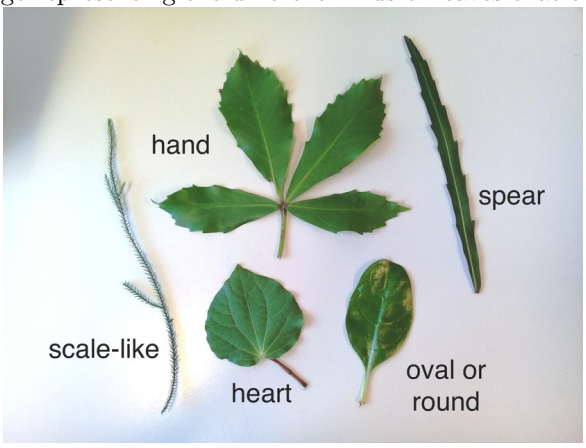
January 28, 2022

Contents

1 Introduction

Image Classification is one of the most fundamental tasks in computer vision. It has revolutionized and propelled technological advancements in the most prominent fields, including the automobile industry, healthcare, manufacturing, etc. There are estimated to be nearly half a million species of plant in the world. Classification of species has been historically problematic and often results in duplicate identifications.

Figure 1: Image representing the different kinds of leaves that exist in nature.



1.1 Problem Formulation

The objective of image classification is to identify and portray, as a unique gray level (or color), the features occurring in an image in terms of the object or type of plants based on their leaves. The extracted features actually represent the shape, margin, and texture of the leaves. Thus, we are tasked with classification of thirteen plant species based on their leaves. Therefore, the objective of this body of work is to use binary leaf images and extracted features, including shape, margin and texture, to accurately identify a number of species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species.

1.2 Paper Outline

This body of work is sectioned in the following manner, first we begin with our proposed solution to the given problem in Section 2. Second, we describe the different procedures which were taken in order to prepare the data for experimentation in Section 3. After that we discuss the experimental setup in Section 4. Next, we discuss the experimentation that was conducted in Section 5.

Finally, we conclude this work by talking about potential areas of improvements in Section 6.

2 Proposed Solution

Countless number of solutions can be adopted and has been proposed for solving many classification problems. The solution that we propose comprises of three main parts: the preprocessing stage, the training stage, and finally the testing stage.

Since the main purpose of this study is to come up with a solution to the plant classification problem, we have decided to come up with two solutions and pit them against each other in an attempt to compare them to see which of the two is best to be used as a solution for this problem. Thus our solution can be described as follows:

- Preprocessing – within the preprocessing stage comprises of first loading the data from their corresponding directory, then resizing the input images into a user specified size, after that we convert the images into a single channel – otherwise known as gray scaling, then we either apply the *Histogram of Oriented Gradients* (HOG) [2] or the *Local Binary Pattern* (LBP) [1] as feature extraction methods. The HOG and the LBP methods were selected in an attempt to: (1) first extract the features from the images, and (2) reduce the size of the input image dimensions to a more manageable size. The final step of the preprocesssing stage is the data splitting.

- Training – within the training phase, we have decided to apply the Support Vector Machine algorithm for multi-class classification, where the number of classes are the given classes inputted by the user.

- Testing – within the testing phase, we have decided to use the Top-1 accuracy measurement – which measures the accuracy based on the top first prediction – as a classification accuracy.

3 Dataset Preparation

Dataset selection is an important step which has the capability to boost the success of the model or render it unusable. The dataset which we have used is a given dataset set in stone, and therefore this step has already been taken care of. The selected dataset is called the *LeafSnap* dataset, which is publicly available on the original *LeafSnap* website. The dataset comprises of $23,147$ lab taken images, categorized under 185 distinct tree species classes.

Data importing consists of loading the images into the image variable which will hold the leaf images to be processed. For each class provided by the user, the class values are distributed based on the enumeration values of the provided classes, i.e., if the algorithm is provided with the following classes:

"abies_concolor", "ginkgo_biloba", and "acer_campestre", the corresponding class values will be distributed as follows: $y = \{1, 2, 3\}$. The following code has been used to load the image inputs into our testing environment:

```
######################### LOAD DATA ##########################
imgs = []
X, y = [], []

# load the images
for c in classes:
    for file in glob.glob(base_dir + c + "\*.jpg"):
        imgs.append(cv2.imread(file))
        y.append(classes.index(c))
```

After loading the data into our testing environment, the preprocessing process begins. The preprocessing process begins with iterating over each input instance and resizing it based on the user defined sizes. After the resizing of the input image, the three channel images are then reduced to a single channel describing the opacity of the pixel which ranges from $[0, 255]$. After the gray scaling process, the algorithm then decides whether to apply the HOG or the LBP feature extraction methods which is selected by the user. Finally, after the feature selection, all of the feature vectors obtained for all of the images are then stored into an input variable X. The following code is used to implement the aforementioned preprocessing methodology which was used in our experimentation:

```
######################### PREPROCESSING ######################
# preprocess the images
for img in imgs:
    img_resized = cv2.resize(img, (width, height),
                             interpolation = cv2.INTER_AREA) # resize
    img_gray = cv2.cvtColor(img_resized,
                            cv2.COLOR_BGR2GRAY) # convert to gray scale
    # extract features
    if apply_hog:
        img_features = hog(img_gray,
                           orientations=9,
                           pixels_per_cell=(14, 14),
                           cells_per_block=(2,2),
                           block_norm='L2-Hys')
    else:
        img_features = np.zeros((height, width), np.uint8)
        for i in range(0, height):
            for j in range(0, width):
                img_features[i, j] = lbp_calculated_pixel(img_gray, i, j)
    X.append(img_features)
```

4 Experimental Setup

In order to implement and study the proposed, multiple technologies were used. This section begins with the description of the hardware setup upon which the experiments were conducted in Section 4.1. Then a description is given on the software setup upon which the experiments were conducted in Section 4.2. Finally, A descrption of the experimentation paramers are discussed in Section 4.3

4.1 Hardware Setup

Since the work is not hardware focused, not much relevance was put into the hardware setup. Below is a list of the hardware that was used during the experimentation process

- Central processing unit: Intel Core i5-8750H with a base clock speed of 2.80 GHz and a turbo clock speed of 2.81 GHz.
- Random access memory: 8.0 GB.

4.2 Software Setup

The software setup is a bit more complex. Below is a list of the software that was used during the experimentation process for the proposed approach

- Operating system: Windows 10 version 20H2, x64-based processor.
- Integrated development environment: Jupyter Notebook version 6.4.6.
- Distribution: Anaconda navigator version 1.10.0 with conda version 4.9.2.
- Programming language: Python version 3.8.3

The Anaconda distribution was chosen to manage the Python environment and library versioning of the experimentation that were conducted. Below is a list of all of the Python libraries that were used for the making of the code for the experiments that were conducted

- NumPy – this is the library that was used extensively for handling the lists, vectors, and matrices.
- OpenCV – this library is used to handle the image loading, grayscaling, and resizing.
- Sci-Kit Learn – this library was used to split the dataset into training testing samples, as well as learn the SVM model.
- Sci-Kit Image – this library was used to apply the feature extraction HOG model to the input instances.
- Matplotlib – this is the library that was used to plot the graphs and visualize the data.

4.3 Experimentation Parameters

The user parameters within our experimentation environment have been selected and remained constant throughout the entirety of the experimentation process. The main parameters which the user can select are the following: the classes which the algorithm will learn to classify, the weight and width of the images to be resized to, and finally whether the algorithm will apply the HOG or the LBP feature extraction methods. The code describing the constant parameters used within out experimentation can be found below:

```
######################### USER INPUTS #########################
# directory of the database
base_dir = "C:\\Users\\ILAA\\dataset\\images\\lab\\"
# list of classes
classes = ["abies_concolor", "ginkgo_biloba", "acer_campestre",
           "acer_pensylvanicum", "aesculus_pavi", "ailanthus_altissima",
           "amelanchier_laevis", "betula_jacqemontii", "carya_tomentosa",
           "chamaecyparis_pisifera", "staphylea_trifolia",
           "phellodendron_amurense"]
width, height = 500 , 500 # width and height of the image
apply_hog = True # if true apply hog otherwise apply lbp
```

An additional dataset analysis has been done in an attempt to understand the number of classes present within each class values. In Figure 2, the class values are represented on the y-axis whereas the number of images is presented on the x-axis. We can conclude from this graph that the classes chosen for our experimentation follow a near constant destribution which is the preferred kind of destitution which we would like to work with so that the model does not form any biases towards specific classes which are more populated.

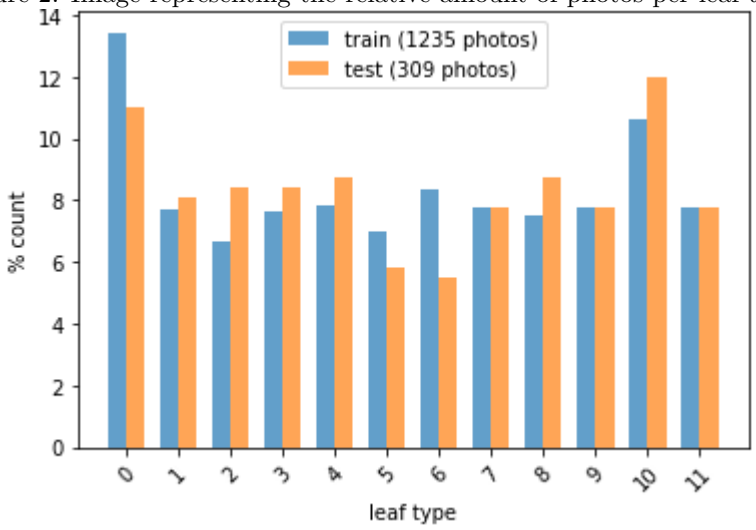
The final stage of the preprocessing is the data splitting. The given split which was decided was a 80% split for the training data, and a 20% split for the testing data. The following code has been used for the data splitting within out testing environment:

```
######################### DATA SPLITTING #########################
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    shuffle=True,
    random_state=42,
)
```

We do note that the implementation of the LBP feature extraction algorithm has been done from scratch where the following code has been used within our experimentation:

```
######################### LBP INITIALIZATION #########################
def get_pixel(img, center, x, y):
```

Figure 2: Image representing the relative amount of photos per leaf type.




```
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1

    except:
        pass

    return new_value

# Function for calculating LBP
def lbp_calculated_pixel(img, x, y):
    center = img[x][y]

    val_ar = []
    val_ar.append(get_pixel(img, center, x-1, y-1)) # top_left
    val_ar.append(get_pixel(img, center, x-1, y)) # top
    val_ar.append(get_pixel(img, center, x-1, y + 1)) # top_right
    val_ar.append(get_pixel(img, center, x, y + 1)) # right
    val_ar.append(get_pixel(img, center, x + 1, y + 1)) # bottom_right
    val_ar.append(get_pixel(img, center, x + 1, y)) # bottom
    val_ar.append(get_pixel(img, center, x + 1, y-1)) # bottom_left
    val_ar.append(get_pixel(img, center, x, y-1)) # left

    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
```

```
    for i in range(len(val_ar)):
        val += val_ar[i] * power_val[i]

    return val
```

5 Experimental Results

The experimentation has been conducted in an attempt to compare two feature extraction methods, the HOG and the LBP feature extraction algorithms, and analyse the obtained results in an effort to reach a certain "winner". The first phase of the experimentation is the training phase where an SVM model was trained over the preprocessed data. The following code was used to train the SVM model:

```
######################### TRAINING #########################
svm = svm.SVC(kernel='linear', random_state=0)
svm.fit(X_train, y_train)
```

The second phase of our experimentation is the testing phase, which was implemented using the following code:

```
######################### TESTING #########################
y_pred = svm.predict(X_test)
acc = 100*np.sum(y_pred == y_test)/len(y_test)

print("Accuracy Linear Kernel:", acc)
```

The resulting Top-1 accuracy rate is returned for both the HOG and the LBP feature extraction methods. First we begin with displaying the accuracy results obtained from the HOG feature extraction method, which resulted with the following accuracy rate:

```
Accuracy Linear Kernel: 98.82462793151521
```

The following Top-1 accuracy rate is the obtained accuracy rate from using the LBP feature extraction method:

```
Accuracy Linear Kernel: 97.73462783171522
```

Based on both of the results displayed above – namely the Top-1 accuracy rate for the HOG and the Top-1 accuracy rate for the LBP feature extraction algorithms, we can conclude that the HOG method for feature extraction works better for the tree classification problem by using tree leaf images as an input.

6 Further Discussion

Within this body of work, we were able to implement and compare two feature extraction methods for the given tree classification problem. The two methods

which were used for comparison are the HOg and the LBP feature extraction methods. The resulting conclusion made based on the experimentation is that the HOG feature extraction method is more superior than the LBP feature extraction method for the given tree leaf classification problem.

However, we do recognize that other feature extraction methods could have been used and implemented for the purposes of this study, e.g., Hu Moments, Run-Length Matrix Features, etc... Another fact which we recognize is that we have selected the SVM model as a learning model to be used for the tree leaf classification problem, therefore we think that it is worth testing out a different classification model, e.g., k-Nearest Neighbors, Decision Trees, Neural Networks, etc... But since our obtained results were satisfactory enough for the classification problem, we consider this body of work to have reach its goal and purpose.

References

[1] Ruaa Adeeb Abdulmunem Al-falluji. Dme detection using lbp features. *International Journal of Computer Applications*, 148(8), 2016.

[2] Hasan Demir. Classification of texture images based on the histogram of oriented gradients using support vector machines. *Istanbul University Journal of Electrical & Electronics Engineering*, 18(1):90–94, 2018.