ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐẠI HỌC QUỐC GIA TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN KHOA HỌC MÁY TÍNH

NHẬN DẠNG

ĐỒ ÁN 1: PATTERN RECOGNITION
PLANT DISEASES ON LEAF

Sinh viên thực hiện: Nguyễn Thế Hoàng (MSSV: 2012 0090)

Giáo viên phụ trách: Lê Thanh Phong - Lê Thành Thái

BÀI TẬP MÔN HỌC - NHẬN DẠNG

HỌC KỲ II - NĂM HỌC 2022 - 2023

1 Problem analyze

1.1 Objective

From a given training dataset includes images of leaf, we need to build a model that can recognize the status of that plant by observing images of leaf. There are 4 status (or categories) that a plant in a picture can be, including:

- Healthy
- Multiple diseases
- Rust
- Scab

1.2 Type of solution

The model will be run on a test set, in offline mode for grading and studying purpose. The model is trained using supervised learning, with batches of sample (offline) and based on model approach.

1.3 Measure performance

As often used in classification problems, we will use these metrics to measure the performance of the model:

- *Confusion matrix* to know the number of true/false classification for each category.
- *Precision score*
- *Recall score*
- *F1 score* for overall performance

1.4 Way of implementation

We will use *scikit-learn* to implement Support Vector Machine (SVM) as the main classifier, and self-implementation for PCA model. We also use *numpy, pandas* to store the numerical and sample while training and predicting, *PIL.Image* library for image processing, and *matplotlib.pyplot* for illustrating images and diagrams.

1.5 Running program

The folder structure of the submitted program is as followed:

- *src*: directory contains source code to run. Includes:
  - *data_ insights.ipynb*: only for overviewing given dataset.
  - *main.ipynb*: main execution files.
- *doc*: directory contains this report file.

	healthy	**multiple_diseases**	**rust**	**scab**
Number of intances	516	91	622	592
Ratio (%)	28.66	5.05	34.54	32.87

Bảng 1: Distribution of each category in the training sample

- *data*: directory contains dataset. It includes the given dataset from the teacher, *submission.csv* contains result on test set, and other saved numpy array under .npz format. In the folder named *images*, there must be all and only images from train and test set.

To run the program successfully, one need to keep the folder structure as described above, even in Colab environment. Run the *main.ipynb* to preprocess the data, train the model and predict on the test set.

2 Data analyze and data processing

2.1 Given dataset

The given dataset includes:

- *Folder of images*: contains all the training and testing images. There are 1821 images for each training and testing set.

  The image is in *.jpg* type. Each has size 2048×1365 in pixels. The image also use RGB channel color format. So a numpy array contains an image has shape $(1365, 2048, 3)$.

- *train.csv* List of name of train samples and one-hot vector encoded respects to each instances.

  After observing training data by running *data_ insights.ipynb* file, we can conclude that:

  - All 1821 training instances have valid one-hot encoded (no null, each image belongs only to 1 category).
  - Number of training intanses of each category is expressed in table 1.
  - Most of the class in training set has same number of instances, except for the class *multiple_ disease* which only has 91 instances. This can cause difficulty for the model to recognize images belong to this class.

- *submission.csv* List of predictions for the test set. There are 5 columns. The first one is the name of image in the test set, the remaining columns are one-hot vector encoded with 4 categories, same as training data.

2.2 Data preprocessing

Since the models we will use are sensitive to the training sample, we need conduct following techniques to clean and transform the dataset, so that we can maximize the performance of the model.

2.2.1 Resize images

The original size of images is 2048×1365 in pixels. If we read all the images in the training set and conduct training the model, we will not have enough memory (it can easily reaches to $2048 \times 1365 \times 3 \times 1801 = 18$GB). So we reduced all the images to the common size 640×374 in pixels, so we only need about 400MB when load all the training images to the memory for training model. The images are read and contain in numpy array type.

2.2.2 Transform to gray sclae

Although the size of images is much more smaller, since each pixel need 3 values of RGB to decode a color value, the number of features are still big. So we transformed all images to the gray scaling. It means each pixel now only need a value between $(0, 255)$ to decode information. As a result, the images loose all information of color (which is quite bad since some diseases only can be deteced through color information), but they still keep information of contrast, pattern.

Now each image (or training sample) have $640 \times 374 = 239360$ features, with value from 0 to 255. An image contained in numpy array type has shape $(374, 640)$.

2.2.3 Histogram equalization

By using histogram equalization, we can even make the images even higer in contrast. This transform flattens the gray level histogram of an image so that all intensities are equally common as possible. As a result, the image intensity is normalized before futher processing and also a way to increase image contrast, the details of the dark regions appear clearly (which is quite important in some diseases with darker region on the leaf).

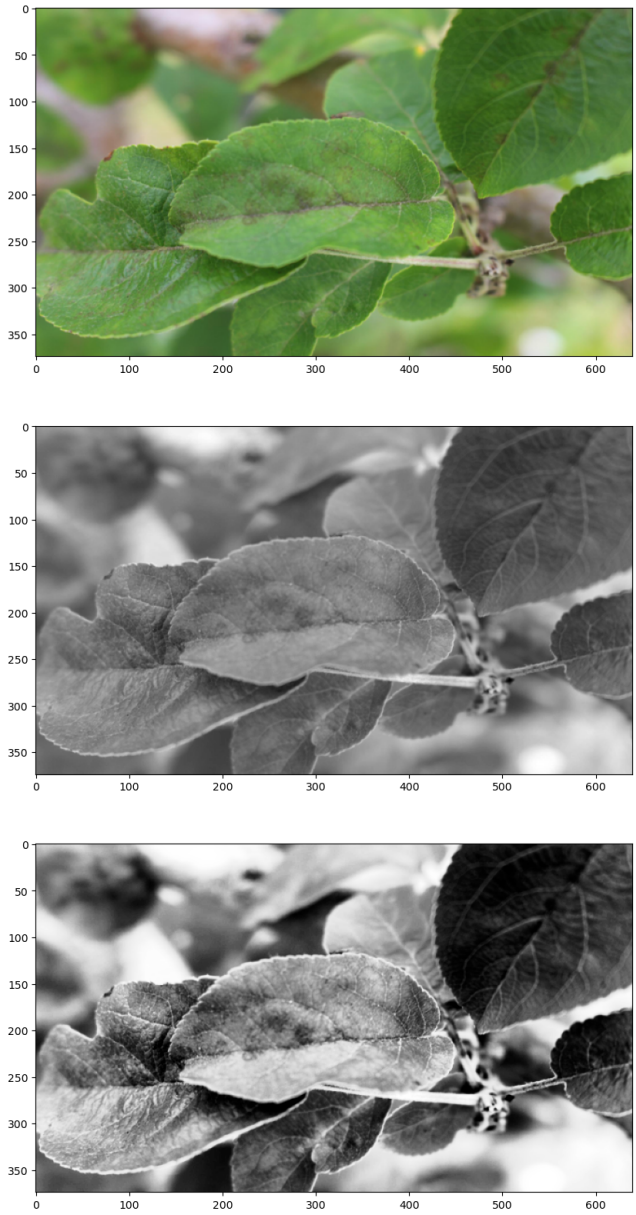
The transform function we used, in this case, is a cummulative distribution function (cdf) of the pixel values in the image. After that all the pixel values are normalized to between 0 and 1 [2]. We can see the result after above transformations in diagram 1.

2.2.4 Feature extraction - Dimensionality reduction: Principal component analysis (PCA)

As the number of intances training are much smaller than the number of features (1801 compared to 239360), to avoid the *curse of dimension* and speed up the training time of Support Vector Machine (SVC) as the classifier, we need to project each image to a smaller manifold. Here I tried using PCA for this purpose. The implementation is as descbribed in [2].

Before conducting PCA, we need stack all training images (after aboved trasforming steps) all together in a numpy array. Additionally, we will flatten all the images to 1D array.As a result, we now have a numpy array has shape $(1801, 239360)$. This will be used through all following training steps. So we have 1801 training samples, each has 239360 features.

After conducting PCA techniques (with compact trick used as the number of dimension is so much higher than number of data), the shape of array becomes $(1801, 30)$. So by using PCA, we have lowered the number of features from 239360 to just only 30. These 30 components have the highest variance to differentiate images in different category, while maintaining local group for images in same class. From now on, we can project every

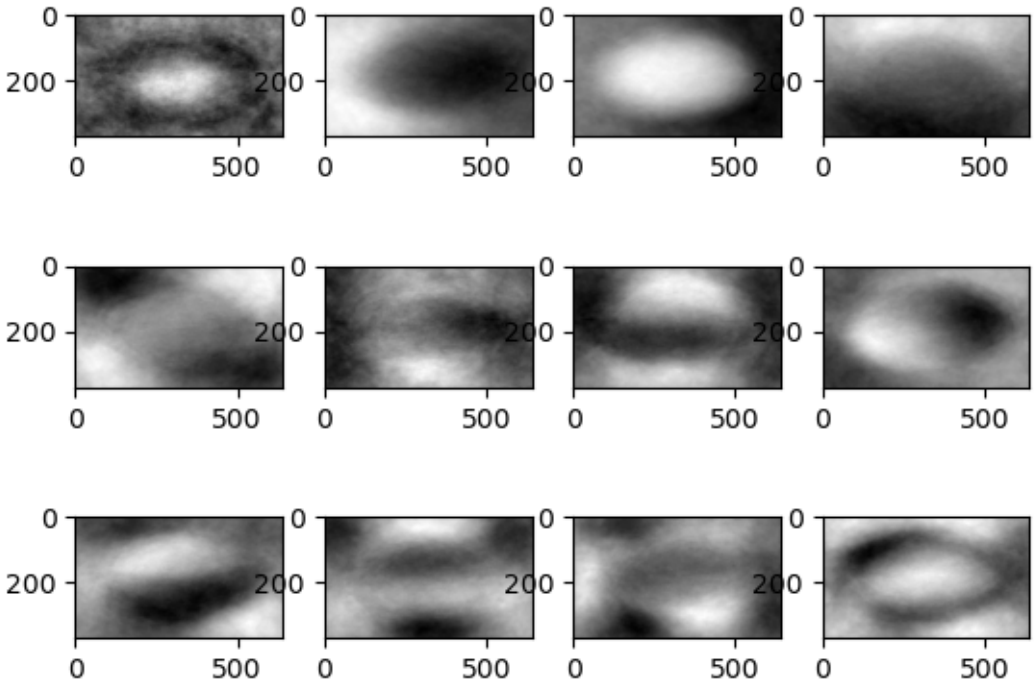


Hình 1: From top to bottom: Original image with RGB channel; Image transformed to gray scaling; Image in gray scaling and histogram equalization

image on this basis to have a "point"in 30-dimension space. By using a classifier, we can classify these point.

2.2.5 Standadize

The last step in the preprocessing data pipeline is standadizing data. Since we use SVM as a classifier, we need to make every pixel value in the training sample have means 0 and unit variance. The standadize process is computed on each feature of the image on the whole set.



Hình 2: The image presentation of: the mean of whole dataset (top left) and 11 "eigen-leaf"sorted in descending order of variance

Score (Average micro)	**Linear SVM**	**RBF Kernel Nonlinear SVM**
Precision	0.3703	0.403
Recall	0.3703	0.403
F1	0.3703	0.403

Bảng 2: The performance score of different type SVM

3 Training model SVC and experiment result

We use two types of SVM for comparing performance: the Linear SVC and the Nonlinear SVC (here we use RBF kernel type). For each type, we manually fine-tune the parameter to have the best performance as following:

- Linear SVC:
  - $C = 1$
- RBF kernel Nonelinear SVC:
  - $\gamma = 0.1$
  - $C = 0.9$

With above parameters, we get the following performance score with 3-cross-validation technique in table 2.

The low performance score can be due to the removal of color information from the images (since the classifier can think that the dark region is always a sign of some diseases.

We may improve by using grid-search technique, or conduct PCA with color version. This will be also the further work we may do to improve the performance of the model.

4 Conduct testing

The images, data of testing set are also preprocessed with all mentioned steps in section 2. The result has been saved to the file named *submission.csv*.

Tài liệu

[1] Géron Aurélien. *Hands-on Machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. English. 2019. ISBN: 9781492032649 1492032646.

[2] Jan Erik Solem. *Programming Computer Vision with Python*. O'Reilly Media, Inc., 2012. ISBN: 9781449316549.