Machine Learning Engineer Nanodegree

Capstone Project

Machine Learning with Small Datasets

Performance comparison for Neural Networks and Principal Component Analysis

Alexander Langer

12th February 2017

I. Definition

Project Overview

The objective of this exercise to identify the species of a plant by processing the image of its leaf through a machine learning algorithm that will output a prediction about the type of leaf it has been given.

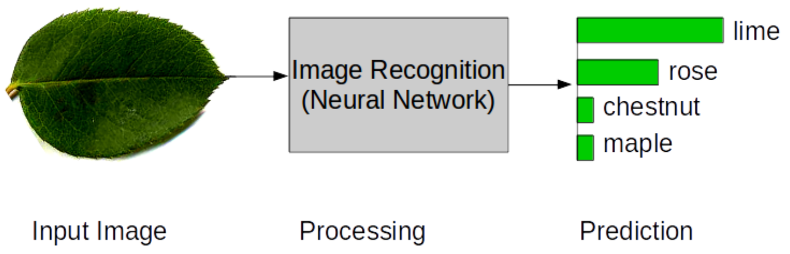


Figure 1 Process Design

The necessity to address this subject is given through global trends in the field of agriculture. As the University of California, Davis summarizes[1] agricultural efficiencies must increase to meet population growth projections of 3-5 Billion until 2050. A key lever will be to lower the units of inputs per unit of output. Foremost, the volume of fertilizers and pesticides must be dramatically reduced to maintain soil and plant vitality.

Image recognition through machine learning can be a key contributor to this challenge since it

1 http://asi.ucdavis.edu/programs/sarep/about/what-is-sustainable-agriculture

enables resource conscious processes in weed elimination[2] or, as currently investigated by the startup PEAT: plant disease detection[3].

The approach taken by this startup is to utilize a neural network architecture to identify infected crops for diseases known to their algorithm. Since farmers, foremost in Germany and India utilize the algorithm as a smartphone app, they also submit image data to the startup which in return supports further learning for their application.

Due to the variety of plants, approximately half a million globally, classification of infected crops or weed is prone to errors. If agricultural output is to increase while pesticide usage is to decrease, reliable image classification by machine learning can be a key contributor.

This paper outlines this classification challenge by deploying two distinctively different supervised learning algorithms – a neural network and principal component analysis (PCA), against images of leaves that belong to one of 99 species. The underlying dataset has been retrieved from the UCI Machine Learning Repository and Kaggle[4]

Most prominent work on PCA and image recognition has been performed by MIT Scientists[5] on an image sample of a few hundred images – the so-called Olivetti faces dataset. Most remarkably, this research has not only been conducted in 1991 but also addressed subjects such as neural network implementation and image localization.

Since our underlying dataset is very small, successful results might not be obtained through the neural network and will thus also be attempted with PCA strategies as well. A discussion comparing the results will justify varying application scenarios for each approach.

My personal ambition to investigate this subject is given due to my professional background in agricultural manufacturing and innovation management.

Problem Statement

Classifying images of leaves to identify their species is a supervised machine learning and classification task. The shape of the leaf will provide distinctive patterns that will become more evident through various preprocessing steps and can then be leveraged through the machine learning algorithm.

2 http://www.bluerivert.com/
3 Http://peat.technology/
4 https://www.kaggle.com/c/leaf-classification/
5 M. Turk, A. Pentland :Eigenfaces for Recognition, 1991

To make the images machine readable, preprocessing such as color transformation to binary colors, resizing and augmentation will take place. The image data will ultimately be transformed into multi-dimensional arrays that can be fed into the algorithms.

Since all image data has been labeled with a plant species type, this aspect must be preserved to train our program and to test its accuracy after training on a dedicated and isolated test dataset.

When referring to the word algorithm, two trajectories will be explored:

1) A deep neural network with dedicated weights and biases for each plant species type. It will have several layers that abstract the input image data further to learn high-level patterns, so called gradients. Since we feed around thousand images into our network, this will be done in small batches while each iteration will adjust the weights and biases in the network to train it against labels – in our case leaf species.

While weights and biases are being trained, and set so that an input image can be identified as belonging to a species, each training iteration will produce a prediction towards the anticipated kind of leaf species. This prediction will be represented with floating point numbers between zero and 1 where each possible species label will obtain a number varying by the degree of certainty the algorithm has about its prediction. This is called the SoftMax function. Afterwards the prediction with the highest likelihood (argmax function) will be referenced against the true prediction and the weights and biases will be adjusted accordingly. This process is called the loss function or optimizer and it is where the actual machine learning takes place iteration by iteration.

The model should gain some robustness and the capability to reproduce prediction when fed new leaf images. I hope to obtain prediction probabilities above 90% while measuring accuracy will be done by comparing predicted labels (e.g. top 3 predictions) vs. Actual labels.

2) Principal Component Analysis aims at extracting reoccurring image features (principal components) from our dataset by searching the n-dimensional image data space for those subspaces most representative for the features of our leaf images. As applied in the eigenfaces project (Turk, Pentland, 1991), Principal Component Analysis can be deployed towards image recognition tasks while only a few image samples are at hand. Due to its core attribute to reduce feature dimensionality it also reduces machine learning model complexity and can those provide robust results with only limited model design requirements.

Fine-tuning the model can be performed with grid-search, evaluating different model parameters automatically.

Metrics

To judge model performance for the principal component analysis, precision, recall and the f-1 score will be deployed. We will thus be able to learn how many times a leaf species was confused for another in terms of a true positive (precision) and false positive (recall) relationship and how many image samples were classified correctly overall.

Precision (P) =	Recall (R) =	F1 Score =
$\frac{\text{True positives}}{\text{(True positives + False Positives)}}$	$\frac{\text{True positives}}{\text{(True positives + False Negatives)}}$	$2 * \frac{P * R}{P + R}$
”…the ability of the classifier not to label as positive a sample that is negative.”	”… the ability of the classifier to find all the positive samples.”	”… a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.”

Figure 2 - PCA Key Metrics

For the neural network, an accuracy function will be deployed as the metric to judge model prediction performance. The metric formula is depicted in a schematic way below and customized for the given one-hot encoded label structure

$$\text{Accuracy} = \ 100.0 * \sum \frac{\text{Argmax(predictions) = labels}}{\Sigma \text{ predictions}}$$

Figure 3- Neural Network Accuracy Function

Since the biases element in our neural network architecture will be trained to learn the transformed array and pixel representation of each leaf, it will enable the network to judge a new image against its fit to one of the pretrained labels and leaf types.

As described, this process will produce statistical probabilities through a SoftMax function that in the overall sum equal one and the trained bias with the highest probability will be our prediction for the most probable leaf type. Comparing this prediction against our ground truth data (actual labels) will yield a match or a wrongful guess. While the argmax function will pick the prediction with the highest probability, we will also be able to learn the percentage of certainty our network had over its prediction. I would label this the actual accuracy of the prediction.

II. Analysis

The underlying dataset for this exercise has been collected by James Cope, Thibaut Beghin, Paolo Remagnino, & Sarah Barman of the Royal Botanic Gardens, Kew, UK. It was made available through the UCI Machine learning repository and Kaggle (https://www.kaggle.com/c/leaf-classification)

Data Exploration and Visualization

The raw dataset is made up of 1,584 images of leaf specimens. There are 99 different specimens with 16 samples each. The distribution of images across the 99 classes is uniform throughout the dataset.

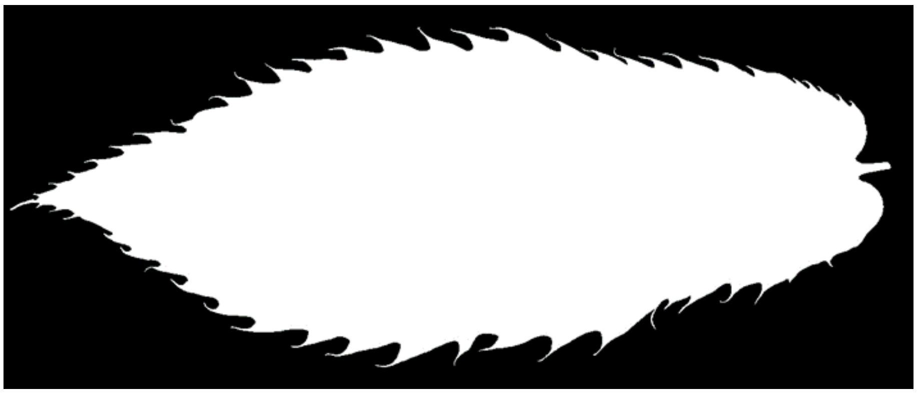


Figure 4 - Sample Leaf Image

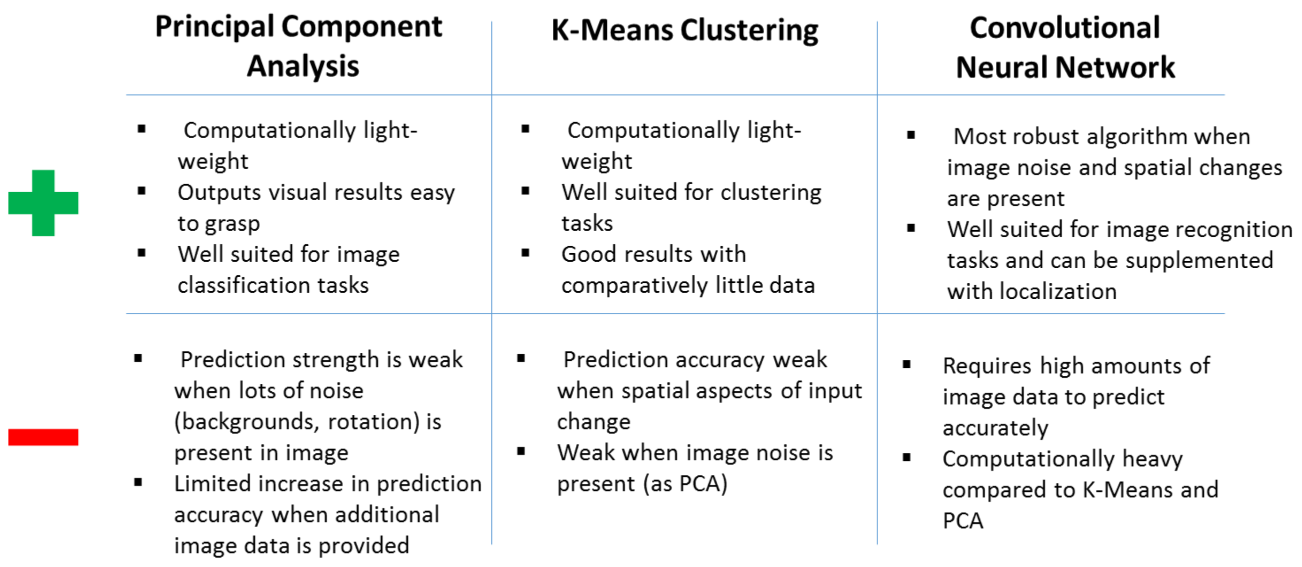
All images have been provided in binary color and inner contours and noise has been removed so that only the outer contour of all leaves has been preserved. Therefore, individual image sizes have been reduced to 5-100 Kb. A sample image of the dataset has been provided above.

While the dimensions of the images given differ largely, some preprocessing will be applied to harmonize the input for the algorithm. Besides the actual image data, csv files have been provided indicating the species of the leaves and some contour data while this data aspect will be omitted for the training of our algorithm.

Algorithms and Techniques

The python based project implementation will borrow from various libraries such as numpy, OpenCV and tensorflow/keras to build a deep neuronal network to learn image patterns and predict the species of a leaf when given a new image. The PCA analysis will mostly be implemented by utilizing the scikit-learn library.

Xinshao et al (Weed Seems Classification Based on PCANet Deep Learning) utilized PCA analysis to solve this natural image recognition task. Mallah et al. (Plant leaf classification, 2013) deployed a K-Means Nearest Neighbor Algorithm but stressed more statistical support data on the image which are also provided with the dataset. I will seek to deploy PCA as well as a Neural network architecture comparable to Goodfellow (Multi Digit Number Recognition, 2014) since deep neural networks have proven exceptionally versatile and suitable to learn image patterns and produce predictions of such kind. While Goodfellow deploys up to 12 different layers for his network, I will seek to obtain results with a recognition accuracy above 90% by including 3-5 layers into my deep network. The only shortfall might be the image sample size, why PCA will be utilized as a second approach for contrasting.



	Principal Component Analysis	**K-Means Clustering**	**Convolutional Neural Network**
➕	▪ Computationally light-weight ▪ Outputs visual results easy to grasp ▪ Well suited for image classification tasks	▪ Computationally light-weight ▪ Well suited for clustering tasks ▪ Good results with comparatively little data	▪ Most robust algorithm when image noise and spatial changes are present ▪ Well suited for image recognition tasks and can be supplemented with localization
➖	▪ Prediction strength is weak when lots of noise (backgrounds, rotation) is present in image ▪ Limited increase in prediction accuracy when additional image data is provided	▪ Prediction accuracy weak when spatial aspects of input change ▪ Weak when image noise is present (as PCA)	▪ Requires high amounts of image data to predict accurately ▪ Computationally heavy compared to K-Means and PCA

Figure 5 - Contrasting of Algorithms in regard of use-case present

The image data that will be fed in batches into the network underwent various preprocessing steps as outlined. It will be provided as a numpy array matching the shape of the networks input layer.

Benchmark

Kaggle competitions addressing the Leaf classification dataset have deployed three main techniques, which overlap with the previously addressed techniques that may be deployed this problem:

PCA Analysis

Algorithms that pursued this approach may have obtained recognition accuracies between 50 and 98% depending on the image and sample sizes given

Neural Networks

Algorithms that pursued this approach may have obtained recognition accuracies above 90% if sufficient image data was provided.Since our approach is mostly constrained by the number of sample images given, I hope to obtain validation accuracies above 50% with either model.

To validate the model results, two different datasets will be maintained: A training and a testing set (Only for PCA) The models will be trained with the first dataset and then tested against its label prediction for the testing set and the actual labels. We will thus derive a figure for accuracy. To get a better sense of prediction accuracy,

III. Methodology

Data Preprocessing

Some preprocessing has already been performed to the provided dataset: All images were transformed to binary color and inner contours of the leaves have been removed. Further preprocessing involved to resize all images to 64x 64 pixels, since heights of up to 1,100 and widths of up to 1,600 pixels were present before.

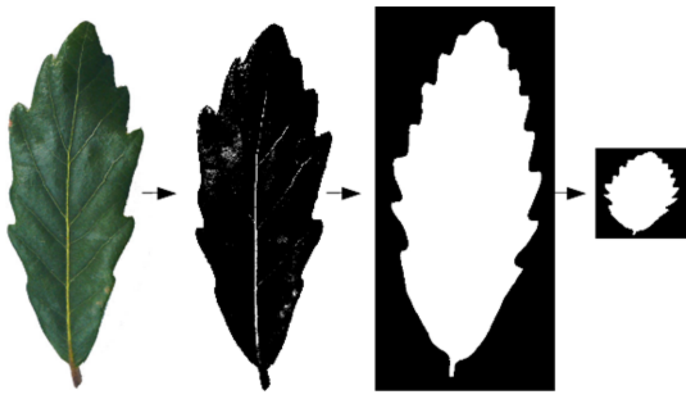


Figure 6 - Image Preprocessing Steps

Labels for the 99 different leaf specimen were provided through csv files. Since the labels were textual, they have been encoded to represent integers from 1 to 99 to feed the neural network with a more readable format. For PCA, the textual labels have been preserved.

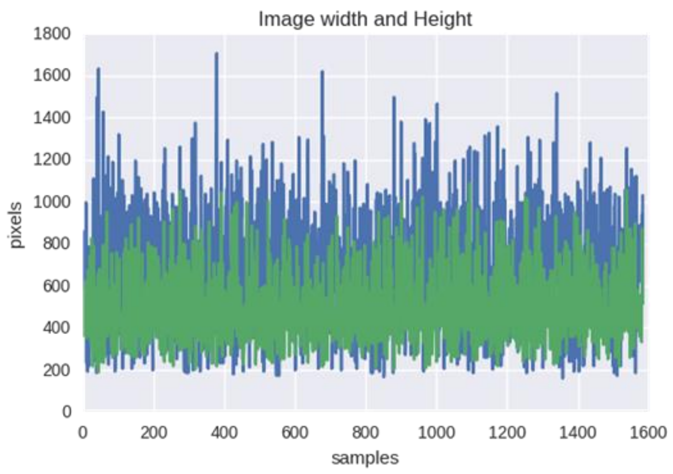


Figure 7 - Image sizes before preprocessing

Reshuffling of the dataset has not been performed, since the distribution of different images and labels has not been in a grouped manner but was perceived well spread. To allow for convenient feeding of the neural network, all image data has been converted to numpy array and has been packaged into a pickle file together with the labels.

The underlying data has also been normalized by dividing all pixel values by 255 (usual rgb color range, also applicable to binary colors). The standard deviation for the n-dimensional image array has thus been reduced from 124. to 0.3 which will allow for more efficient Neural network training.

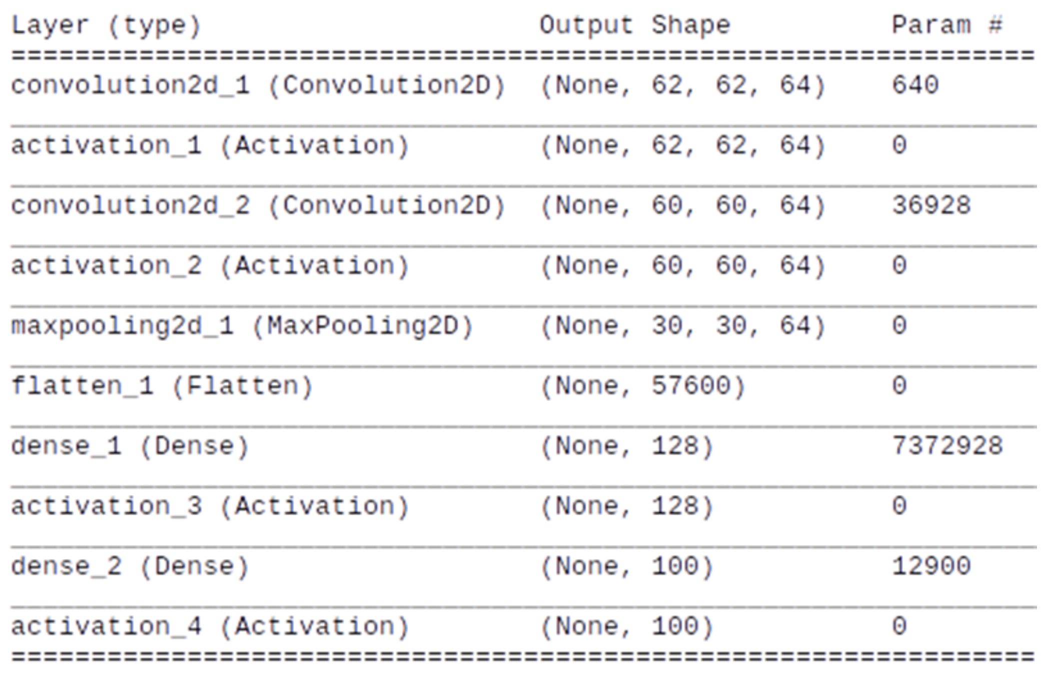
Furthermore, increasing the number of samples has been implemented through KERAS data augmentation techniques.

For PCA, a subset of the images has been preserved for testing. This has been omitted for the Neural Network approach since the dataset has been very small.

Implementation

Neural Network Implementation

Keras together with a tensorflow backend has been chosen to implement the neural network for image recognition. It is fed with a 4-D image tensor (Number of samples, width, height, color channels) and the deployed network architecture is displayed below:



```
Layer (type)                     Output Shape          Param #
================================================================
convolution2d_1 (Convolution2D)  (None, 62, 62, 64)    640
________________________________________________________________
activation_1 (Activation)        (None, 62, 62, 64)    0
________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 60, 60, 64)    36928
________________________________________________________________
activation_2 (Activation)        (None, 60, 60, 64)    0
________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 30, 30, 64)    0
________________________________________________________________
flatten_1 (Flatten)              (None, 57600)         0
________________________________________________________________
dense_1 (Dense)                  (None, 128)           7372928
________________________________________________________________
activation_3 (Activation)        (None, 128)           0
________________________________________________________________
dense_2 (Dense)                  (None, 100)           12900
________________________________________________________________
activation_4 (Activation)        (None, 100)           0
================================================================
```

Figure 8 - Neural Network Architecture

As depicted the input layer is followed by two convolutional layers that each feed into a relu activation function. The stride and kernel each have a size of 3x3 . The border-valid mode is chosen. Afterwards, a 2x2 max-pooling layer is attached followed by another relu activation and another dense-layer + SoftMax activation.

The results feed into a categorical cross entropy function and an adagrad loss optimizer that automatically tunes the learning rate for the model.

As outlined in the data preparation section, a keras data augmentation engine is also deployed while results were not affected by utilizing this technique.

Principal Component Analysis Implementation

For this section, the preprocessed image data has been converted into a 2D-Array (Number of samples, pixels flattened) and the labels have been preserved in textual format.

Since Scikit-Learn has been deployed, the train-test data split function has been utilized where the test set size was set to 0.2.

The Randomized PCA function has then been utilize to conduct the dimensionality reduction and the top 200 principal components (most significant explaining variance) were fed into a sigmoid Support Vector Machine for training.

Then, various hyperparameters were tuned with grid-search and the classifier was fitted to the training data.

As an ultimate step, the trained classifier was deployed towards our remaining test data subset and predefined metrics were retrieved.

Complications during Implementation

Numerous complications occurred during different steps of the project implementation. The most significant ones and the taken contingencies are listed by project section below:

Image Preprocessing

- Sequence of images and labels initially was confused since looping through the folders confused the sequence of filenames. The contingency taken was to have the for loop not read filenames from the folder but to construct them from the csv-file,

- Loading the images into a numpy array originally caused damaging the image data. Expanding the axis on the numpy array after loading the image resolved this,

Neural Network Implementation

- Tensor flow loss function produced "nan" loss values since it attempted to calculate the log on 0 values of one hot-encoded array – This has originally been circumvented by deploying a clipping function and adding some noise values

- Tensorflow implementation was eventually abandoned since the loss function issue was not resolved and it appeared as if predictions were made on the wrong axis of the array.

- Keras has been deployed to resolve the previously stressed Tensorflow implementation issue

Principal Component Analysis

- Prediction of labels was inconclusive due to different array shapes for predictions and true labels. The issue was resolved with various reshaping and intermediate program steps for debugging.

Refinement

Neural Network Refinements

The original network architecture was designed in raw tensorflow while accuracies remained between 1 and 10%. However, various optimization techniques have been deployed:

- Random weight initiation
- Weight and Bias Normalization
- Exploration of various loss functions, optimizers such as adagrad or adam

Since the model did not learn, this approach has been abandoned.

Principal Component Analysis Refinements

Initial PCA results yield accuracies (F1-score) of 0.42. Grid-Search has been deployed for parameter tuning and the following final parameters were obtained:

```
SVC(C=1000.0, cache size=200, class weight='balanced', coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.005, kernel='sigmoid',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Figure – Grid Search Parameter Optimization

A further optimization attempt was to extract a varying degree of eigenleaf-vectors:

- Extracting the top 100 eigenvectors resulted in F1 scores around 0.44
- Extracting the top 150 eigenvectors resulted in F1 scores around 0.4
- Extracting the top 200 eigenvectors resulted in F1 scores around 0.47
- Extracting the top 250 eigenvectors resulted in F1 scores around 0.45

Thus, the option with 200 eigenvectors has been pursued further.

IV. Results

Model Evaluation and Validation

Neural Network Results

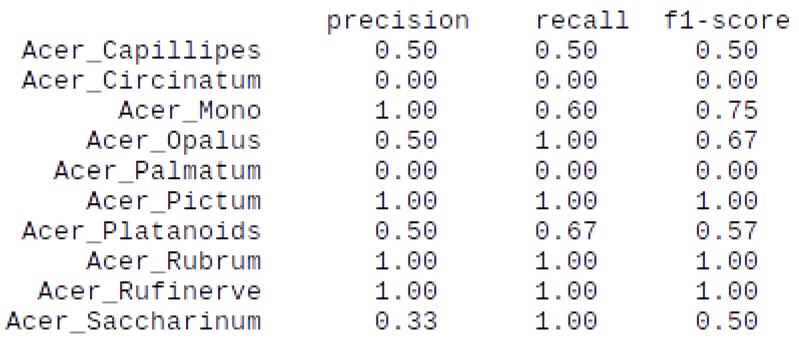
Given the neural network was trained with 990 images belonging to 99 different classes, **accuracy scores of 10% were obtained**. This is significantly below the anticipated score of 80-90% but appears adequate due to the small dataset and the high variety of labels to be predicted.

For proper neural network training, around 1.000 images – 100 times more than were available, should have been deployed to the model

Therefore, it becomes evident that a Neural Network Architecture was inadequate for the given dataset and most the discussion will be led for the PCA-Analysis.

Principal Component Analysis

PCA has yielded F1 scores of 0.47%. Given that we have 99 different labels with rather comparable leaf images, it appears like a very robust result.



```
                   precision    recall  f1-score
   Acer_Capillipes      0.50      0.50      0.50
   Acer_Circinatum      0.00      0.00      0.00
         Acer_Mono      1.00      0.60      0.75
       Acer_Opalus      0.50      1.00      0.67
     Acer_Palmatum      0.00      0.00      0.00
       Acer_Pictum      1.00      1.00      1.00
   Acer_Platanoids      0.50      0.67      0.57
       Acer_Rubrum      1.00      1.00      1.00
    Acer_Rufinerve      1.00      1.00      1.00
  Acer_Saccharinum      0.33      1.00      0.50
```

Figure - Sample Results PCA-SVM

The identification of all true positives over all selected items (precision) also seems rather comparable to the recall score (selected item over all that should have been selected) . Furthermore, very distinctive leaf samples (Acer Rufinerve) had a 100% recall and precision score, suggesting model robustness.

Further evidence in model robustness is given when a varying number of underlying principal components (eigenleafs) has yield comparable results as outlined in the previous refinement section.

Actual predictions are shown below. While the impression is that the true over predicted match is higher than expressed through our 47% F1 score, it is true that the type of leaf family (first syllable) has been predicted more frequently correct than the subgroup (see two Quercus samples)

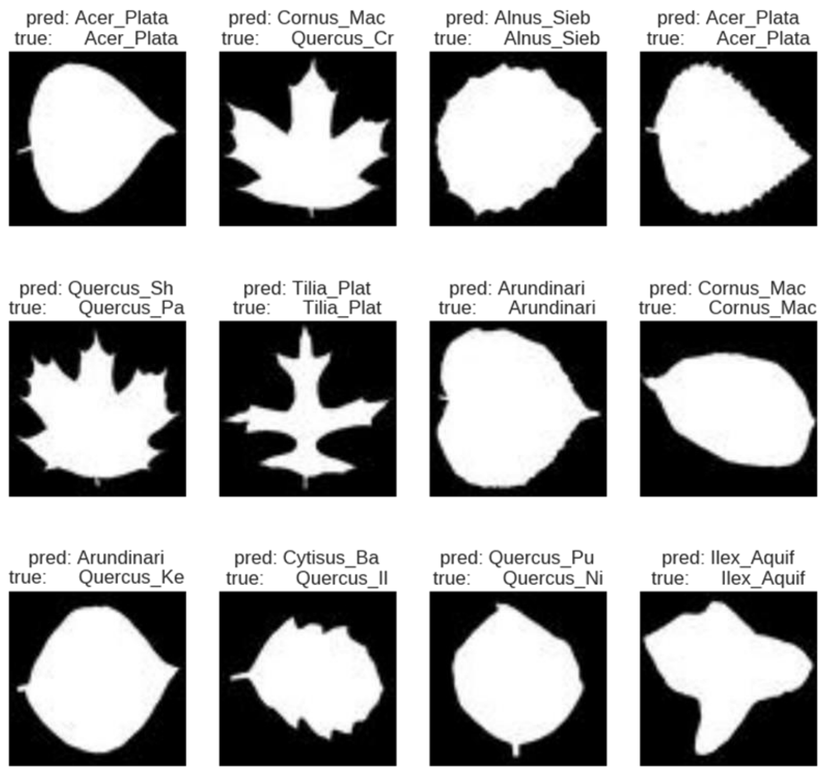


Figure 9 - PCA Results with predicted and actual labels

The model has also been tested on three previously unseen and unlabeled images. While all three predictions were „Quercus_Ellipsoidalis“ - one of the three samples actually has been from that species and another image from the same subfamily. This result seems conclusive with previously obtained F1 scores around 47%.

Justification

As indicated, the neural network approach has fallen short due to small amounts of data – insufficient to properly train a neural network even though data augmentation was applied. Therefore, the accuracy score of 10% is significantly below originally intended scores above 90% but this result is conclusive for the given circumstances.

The PCA analysis has performed significantly better, considering that we have 99 possible image labels with leaf that that would be hard to distinguish for humans, an F1 score of 47% seems very adequate.

The challenge to classify leaf species might thus not have been solved completely, yet more data must be made available first before Neural Networks or other methods might be deployed with a more successful yield.

V. Conclusion

Free-Form Visualization

When considering the main principal components of this analysis, the high overlap of leaf species becomes evident:

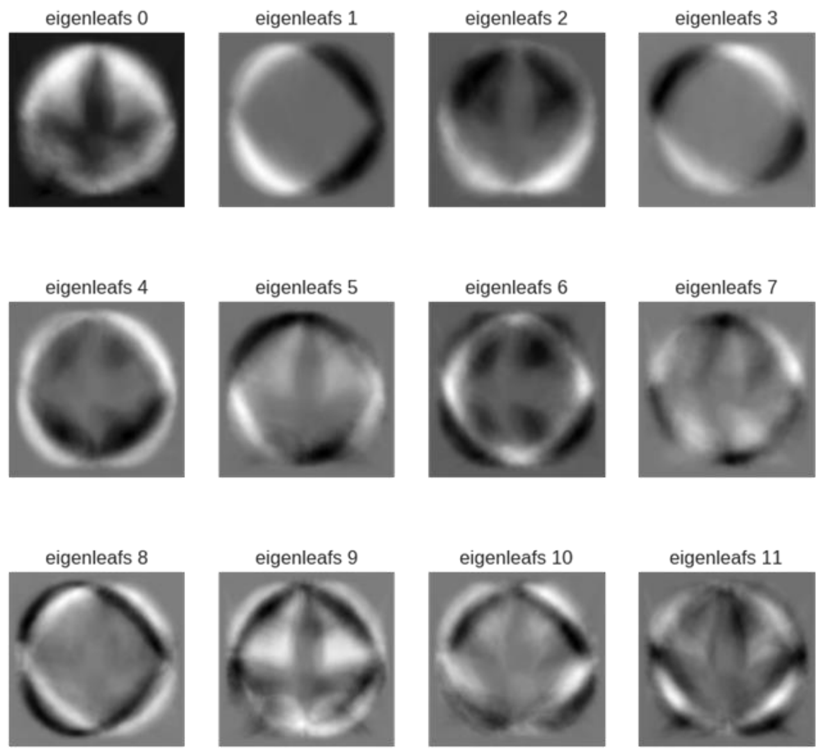


Figure 10 Most Prominent PCA feature extraction

What we can see is the features of leaf image data that explain most of the variance in our dataset – circular shapes and 3-edged star shapes are most prominent. This is conclusive when considering the underlying image data of our leaves.

Since many leaf species have very similar ground shapes, this further explains the strong overlap in principal components and our F1 score of 47%.

Reflection

The underlying leaf classification task foremost was an exercise in preprocessing, since the given data format and normalization highly affects how robust a classifier or neural network can be trained. Especially the latter requires large amounts of data which were not present for the given exercise – only ten images per class were present.

Thus, emphasizing on the PCA analysis provided more promising results, even though these would not be as robust when spatial changes such as image rotations and zooming would be performed.

When setting up the neural network, the KERAS library provided very efficient and robust guidance which rendered lots of the fine-tuning performed in a previously attempted tensorflow implementation obsolete. Results however remained poor with 10% accuracy.

Implementing the principal component analysis was foremost an exercise of three parts:

- Reducing image dimensionality by extracting principal components
- Fitting and training a classifier (Support Vector Machine – Sigmoid Kernel)
- Predicting leaf species with the trained classifier

As testing the classifier on unseen image data has shown, prediction accuracies may vary between 30 and 50% and should thus be improved to solve this exercise adequately.

Improvement

Expanding prediction accuracies for the neural network is foremost an exercise of expanding the amounts of data for the algorithm.

When considering the PCA-SVM classifier results, they may be improved by either combining other methods (ensembled methods) such as random forest or xgaboost to the classification process, Also, other data dimensions such as shape histograms could be utilized and deployed against the same classification methods.

It may thus be interesting to deploy given histogram data to supplement PCA with other ensembled methods – given results in the kaggle data science forum, evaluation accuracies around 95-95% thus may be possible.