APPLICATION OF NATURAL IMAGE PROCESSING TO PLANT SPECIES IDENTIFICATION

A PREPRINT

Scott A. Lewis*
Department of Computer Science
Saint Louis University
St. Louis, MO 63101
slewis3827@gmail.com.edu

May 6, 2019

ABSTRACT

Here, we evaluate the applications of four machine learning models, optimisation strategies and ensembling methods to leaf image data in order to identify the species from which these leaves originate. These models include k-nearest neighbour (KNN), decision tree, random forest, convolutional neural network (CNN), and support vector machine (SVM). We also evaluate hyper-parameter optimisation techniques such as parameter grid searching, and ensembling methods such as bootstrap aggregation (bagging) for a subset of these algorithms. We find that applying bagging to k-nearest neighbours yields one of the highest percent accuracies ($\tilde{9}2\%$), and a 50% reduction in log loss compared to results obtained without bagging. Overall, we find that hyper-parameter optimisation and ensembling in yield the best results and outperform

***K*eywords** Machine learning · Image processing · Plant phenotyping more

1 Introduction

The automation of plant species identification and phenotyping has become increasingly relevant with the resurgence of machine learning applications as a popular means to extract important defining features and process vast amounts of data. Many research groups are now implementing automated mechanical systems to sample images of one or more plant species of interest (Fig 1) in a high throughput fashion [1].

This opportunity for faster phenotype species detection opens up new potential for innovations in agriculture, however, the high level of throughput also introduces the modern problem of a data processing bottleneck. It is an ongoing initiative at many phenotyping centers to develop robust, scalable algorithms that will be able to detect plant image features of interest and use them to accurately make the appropriate classification. While a wide range of algorithms have been employed for this task, it is still unclear how the performance of these algorithms compare with one another in the context of a single image classification problem.

*Corresponding author

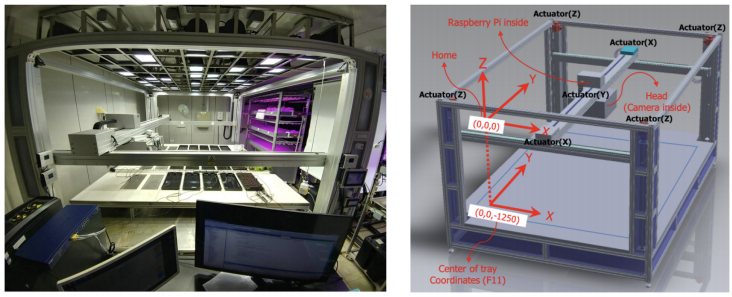

Figure 1: Stationary plant phenotyping machine and schematic of functional components. Images from *Lee, 2018*.

2 Data

The data consist of 16 images each of 99 different plant species obtained from the Royal Botanic Gardens[2] and were downloaded from the University of California Irvine Machine Learning Repository[3]. Each of the 1,584 total images were "binarised" into white leaf shapes against black backgrounds (Fig 2). Each image was also accompanied by a corresponding triplet of extracted features: a *shape* contiguous descriptor, an *interior texture* histogram, and a *fine scale* margin histogram. The three features are each represented as a 64 attribute vector for each leaf sample image. For the following evaluations all data labels were numerically encoded using Scikit-Learn's LabelEncoder() and all images were reshaped with a max dimension of 96 squared pixels. We split the data using stratified shuffling into a train test split of 80% training data and 20% testing data for all model evaluations.

Figure 2: Example training image from the data set.

3 Methods

3.1 K-Nearset Neighbours

The first algorithm used here is the k-nearest neighbours classifier. The basic principle is to take a predetermined number of points, k, closest to the point of data it is attempting to classify, and use those points to predict the identity of that point. The range of hyper-parameters for a classification algorithm is often specific to the data the algorithm is attempting to classify. For this study, we tested the the hyper-parameter k at 3, 5, and 7 in order to survey the best fit for these data. We also implement an ensemble based meta-estimation technique know as bootstrap aggregation, or "bagging" which samples random subsets of the data to fit to multiple instances of a given base classifier before aggregating their individual results. In Table 1, we compare the KNN algorthm with and without bagging to evaluate the effect of the hyper-parameter k, and the influence of bagging on percent accuracy and log-loss. The correlation between percent accuracy and k is inverse, which indicates the model is beginning to overfit at larger values of k.

Model	k	Accuracy(%)	Log-Loss
KNN	3	90.4040	1.5729
KNN	5	89.3939	1.5400
KNN	7	83.3333	1.0251
KNN+Bagging	3	91.4141	0.4115
KNN+Bagging	5	85.8586	0.7432
KNN+Bagging	7	83.8384	0.9037

Table 1: Evaluation of KNN at different hyper-parameters.

3.2 Decision Tree & Random Forest

We implemented multiple decision tree algorithms, at depths 100, 500, and 1000, as well as ensembling multiple decision trees into a random forest model at each depth. In addition to improving the accuracy of the model, ensembling reduces the log-loss by a factor of 10. We see a poitive correlation between depth and percent accuracy in decision trees, while this correlation is inverse in random forests.

Decision Tree A relatively simple model that organises features by making classification decisions based on features that will minimise the entropy and maximise the information gain from each split in the decision tree.

Random Forest Here we make use of a popular ensembling format with decision trees. A random forest builds multiple decision trees from the data, and uses each of the individual decision trees to vote for the most likely classification.

Model	Depth	Accuracy(%)	Log-Loss
Decision Tree	100	66.6667	11.5129
Decision Tree	500	67.6768	11.1640
Decision Tree	1000	70.2020	10.2918
Random Forest	100	92.4242	0.9170
Random Forest	500	89.3939	1.1657
Random Forest	1000	89.8990	1.2993

Table 2: Evaluation Decision Tree & Random Forest at depths 100, 500, and 1000.

3.3 Convolutional Neural Network

The third method we evaluated was a convolutional neural network (CNN). CNNs have been a popular choice for image classification and scale remarkably well to large data sets. The architecture of the CNN used for this classification consisted of a 96 by 96 grayscale input layer, followed by 3 hidden layers of pooling and two dense layers with adjusted dropout at each. A summary of the model and parameters at each layer is shown in Figure 3.

We achieved the best results using a truncated initialisation distribution centered at 0 such that stddev = sqrt(2 / (input-weights + output-weights)) At each layer of activation, we used rectified linear unit activation and optimised with Adam.

With a batch size of 128 and over 1000 epochs our convolutional neural network achieved over 92.8% training accuracy and converged in less than 50 epochs. Training and validation categorical cross-entropy loss scores follow a similar trend converging in less than 100 epochs (Fig 4), while there is sizeable gap in training and validation percent accuracies, indicating a risk of over-fitting in this model after 100 epochs (Fig 4).

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 96, 96, 8)	208
activation_9 (Activation)	(None, 96, 96, 8)	0
max_pooling2d_9 (MaxPooling2	(None, 48, 48, 8)	0
conv2d_11 (Conv2D)	(None, 48, 48, 32)	6432
activation_10 (Activation)	(None, 48, 48, 32)	0
max_pooling2d_10 (MaxPooling	(None, 24, 24, 32)	0
flatten_5 (Flatten)	(None, 18432)	0
dense_13 (Dense)	(None, 1024)	18875392
dropout_9 (Dropout)	(None, 1024)	0
dense_14 (Dense)	(None, 99)	101475
dropout_10 (Dropout)	(None, 99)	0
dense_15 (Dense)	(None, 99)	9900

Figure 3: Summary of the convolutional neural network model's architecture.

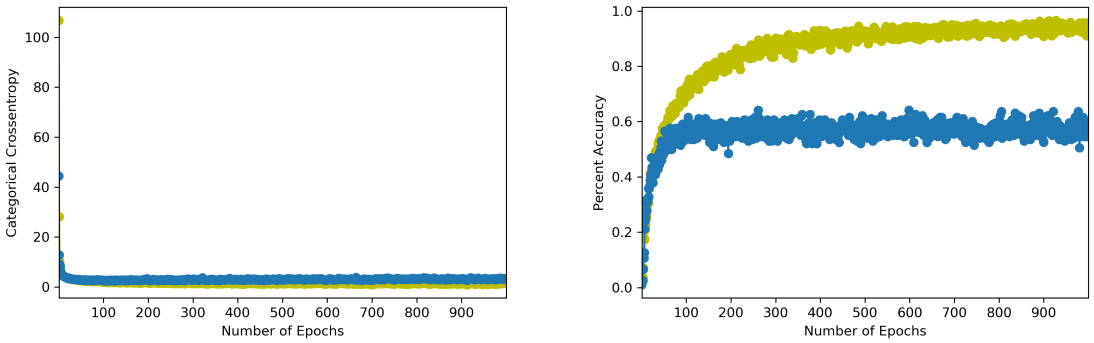

Figure 4: The categorical cross-entropy loss of both training and validation sets converges at <50 epochs.

3.4 Support Vector Machine

Our final evaluation consists of multiple support vector machine implementations, which include a standard SVC, an NuSVC with parameter "nu" that regulates the upper bound of training, and an optimised SVC. We conducted a grid search of "C" values (0.01, 0.1, 1, 10) and gamma values ('auto', 0.01, 0.1, 1) in order to identify the optimal values in these ranges which would result in the best performance.

Model	C/Nu	Gamma	Accuracy(%)	Log-Loss
SVC	1.0	auto	79.2929	4.6107
NuSVC	0.5	auto	85.3535	2.4939
SVC *w/ opt*	10.0	1.0	92.9293	2.3055

Table 3: Comparison of Support Vector Classifiers at default and optimised parameters

4 Results & Dicussion

Four classification algorithms were implemented to be evaluated for their performance at classifying plant species using features extracted from the leaf images. We found that overall the support vector machine that was optimised through a grid search of C and gamma hyper-parameter values had the greatest percent accuracy (Table 3), while k-nearest neighbours with bootstrap aggregation had the lowest log loss (Table 1).

Model	Accuracy(%)	Log-Loss
KNN+Bagging	91.4141	0.4115
Random Forest	92.4242	0.9170
SVC *w/ opt*	92.9293	2.3055
CNN	57.5800	3.2135

Table 4: Best configuration from each evaluated model.

5 Conclusions & Future Work

Surprisingly, the convolutional neural network only achieved a 59% accuracy score on the validation data set, which may be attributed to the neural networks tendency to perform poorly with relatively small amounts of training examples (16 images per species). Based on these findings it is reasonable to conclude that the performance of simple classifiers can thrive with a small number of training examples, while the performance of neural networks may benefit from additional training examples in future studies.

References

[1] Lee, U., Chang, S., Putra, G. A., Kim, H., Kim, D. H. (2018). An automated, high-throughput plant phenotyping system using machine learning-based plant segmentation and image analysis. *PloS one,*13(4), e0196615. doi:10.1371/journal.pone.0196615

[2] Charles Mallah, James Cope, James Orwell. Plant Leaf Classification Using Probabilistic Integration of Shape, Texture and Margin Features. Signal Processing, Pattern Recognition and Applications, in press. 2013

[3] Evaluation of Features for Leaf Discrimination', Pedro F.B. Silva, Andre R.S. Marcal, Rubim M. Almeida da Silva (2013). *Springer Lecture Notes in Computer Science*, Vol. 7950, 197-204