Steven Mills
PHYS247 Final Project
Spring 2020

Project Description

This project is based on data obtained from the Kaggle leaf-classification prediction competition page. https://www.kaggle.com/c/leaf-classification/overview .

Since the competition is over 3 years old, it is not possible to submit the classified/labeled test data in order to obtain a test score. For this reason, I chose to limit my dataset to that contained in the train.csv file – which contained labeled data. The train.csv dataset was divided into train and test subsets for evaluation purposes.

The train.csv dataset contains 990 labeled data. Each labeled data consists of:

- 192-vector
  - 64-vector Shape contiguous descriptor
  - 64-vector Interior texture histogram
  - 64-vector Fine-scale margin histogram
- Silhouette image .jpg file.

figure 1. "78.jpg" – an image of Acer Saccharinum leaf

The goal of the project is to create an image classifier, using a Convolutional Neural Network (CNN), with several Conv2D layers in addition to fully connected layers at the final stages.

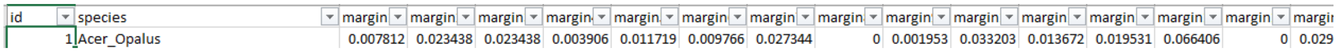
After constructing a CNN classifier using Keras deep learning API (https://keras.io/) in Python, its performance was evaluated using accuracy, precision and recall metrics.

For the purpose of comparison, a classical K nearest neighbors (KNN) model (https://scikit-learn.org/stable/) was also evaluated.

Data Processing and Cleaning

.jpg loading

Leaf images were provided in .jpg files. The images have an id number embedded in the filename. For example, "1.jpg" corresponds to the csv entry with id field of 1.



id	species	margin	margin	margin	margin	margin	margin	margin	margin	margin	margin	margin	margin	margin	margin	margi
1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0	0.001953	0.033203	0.013672	0.019531	0.066406	0	0.029

figure 2. id-to-species association

resizing

The images were converted to uniform sized, 150x150 pixel, grayscale images, using Python PIL.

augmentation

A major challenge for this project is the fact that there are a very small number of images, only 10 per species, and those need to be split into train/test. It is a very small number of images to train the CNN. In order to improve classification with limited training samples, the data set is **augmented** by adding rotated and flipped copies of the images to the dataset.

edge detect filter

In addition to augmenting the images, I chose to filter the images to use only the silhouette outline, to obtain a slightly better result.

species label encoding

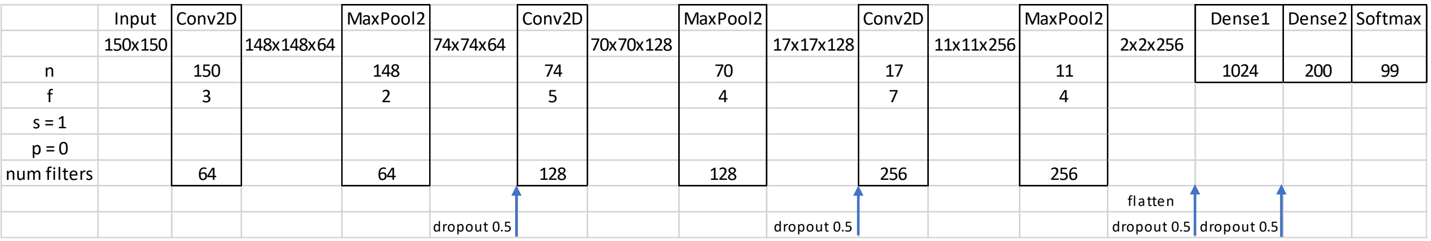
The labels used by the classifier are read from the species column of the train.csv file, by using the id that is embedded in the .jpg file name. One-hot encoding is used to encode the species labels.

CNN Model

CNN architecture as shown below. I based this architecture, on keras example given here: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html .

Architecture

- 3 Conv2D layers + 2 Dense Layers
- 50% dropout between layers, to reduce overfitting
- Rectified Linear (relu) activation used for all layers
- Output layer activation with softmax, size set to number of species, 99
- Adaptive Moment Estimation (adam) optimizer



	Input	Conv2D		MaxPool2		Conv2D		MaxPool2		Conv2D		MaxPool2		Dense1	Dense2	Softmax
	150x150		148x148x64		74x74x64		70x70x128		17x17x128		11x11x256		2x2x256			
n		150		148		74		70		17		11		1024	200	99
f		3		2		5		4		7		4				
s = 1																
p = 0																
num filters		64		64		128		128		256		256				
													flatten			
					dropout 0.5				dropout 0.5				dropout 0.5	dropout 0.5		

figure 3. CNN Architecture

CNN Result.

72 % accuracy

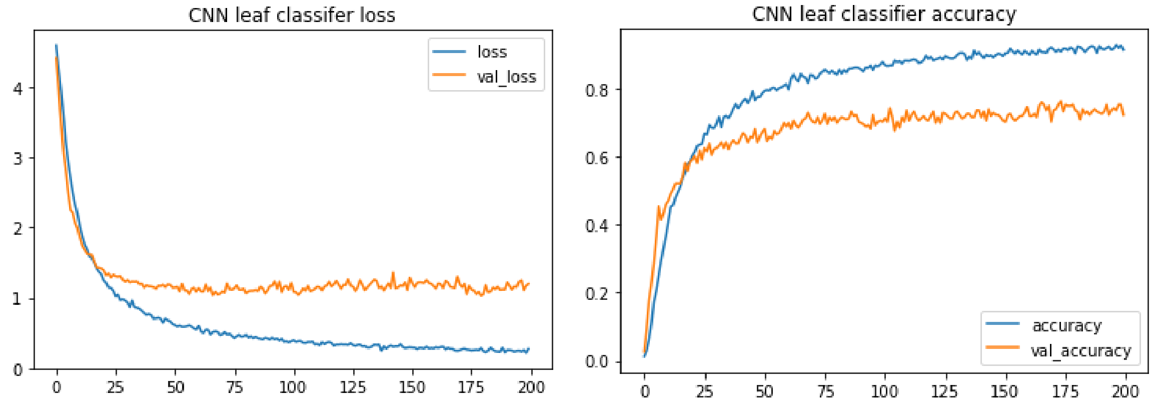


figure 4. CNN loss, accuracy plots of 200 training epochs

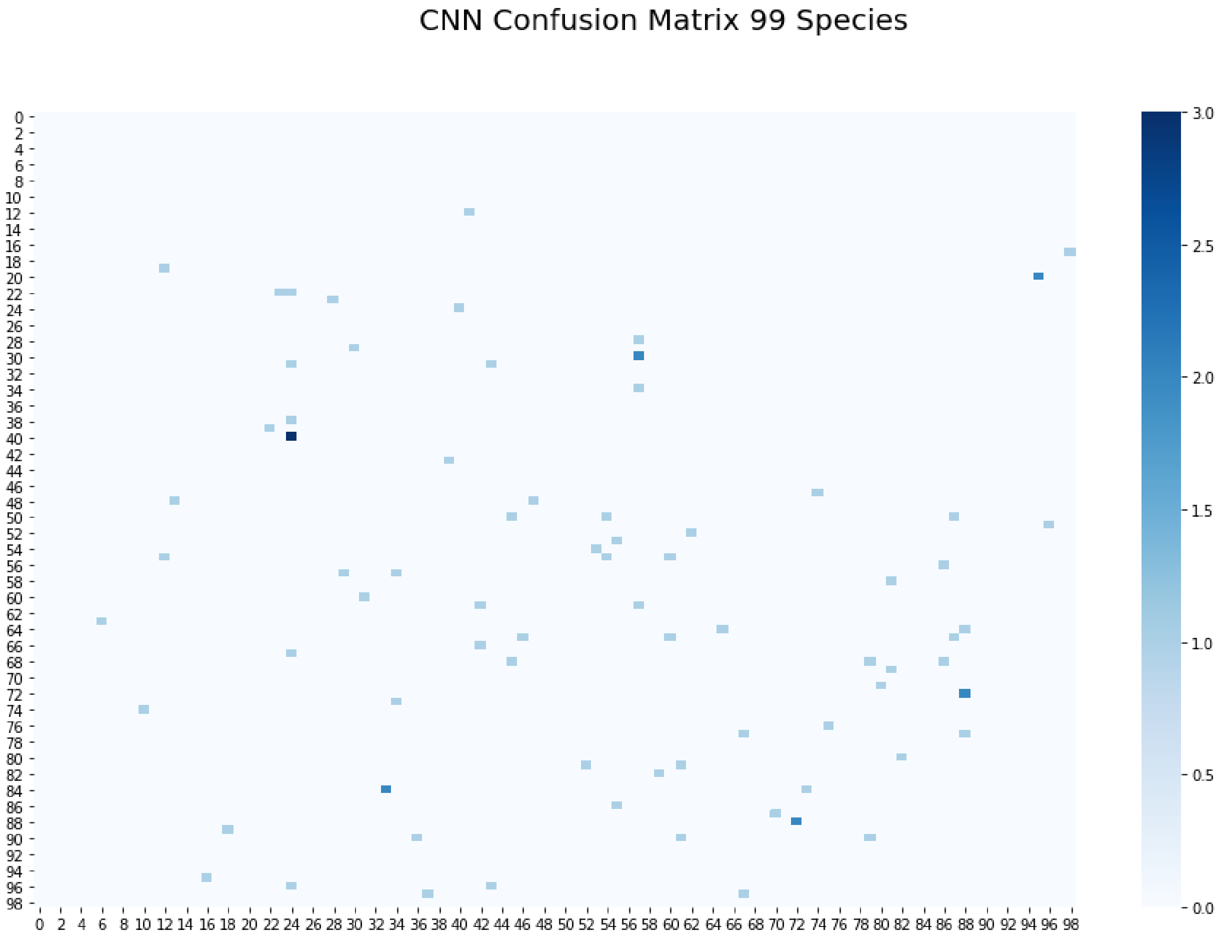
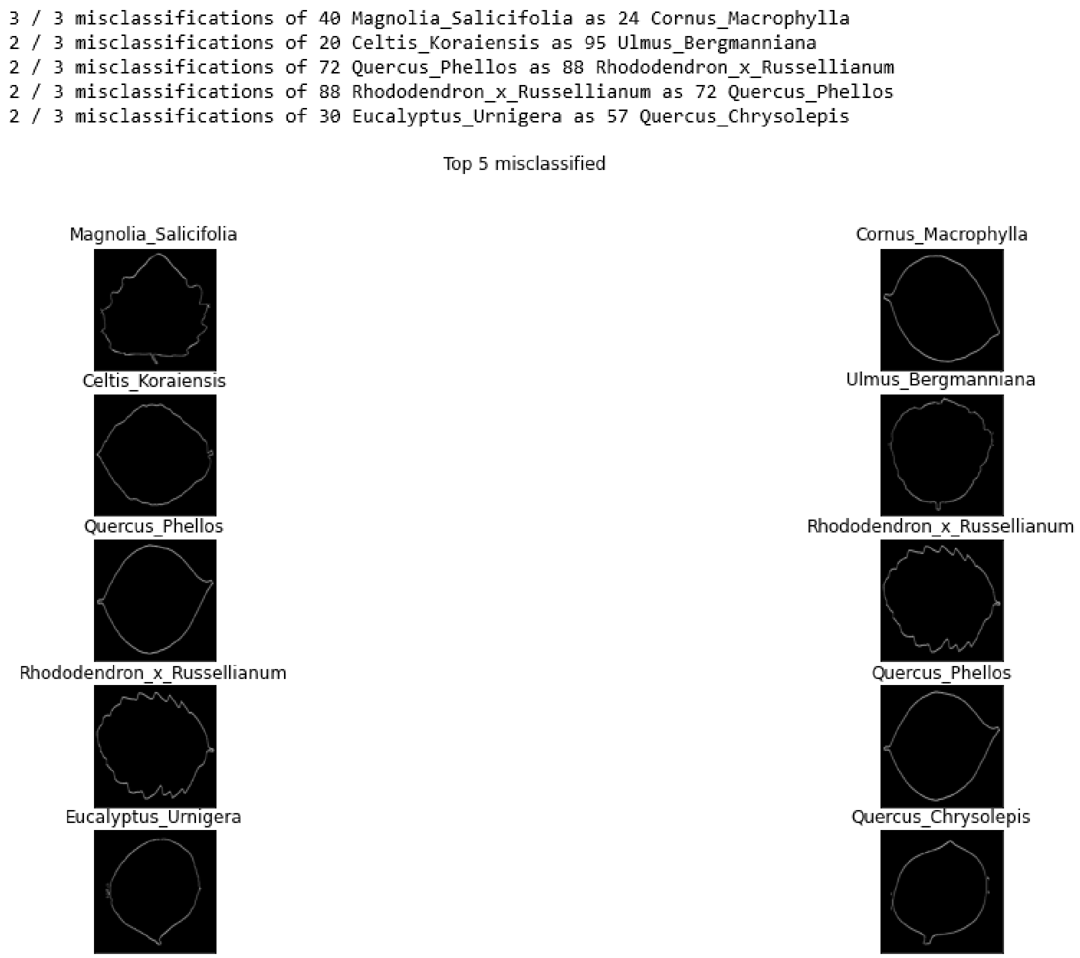


figure 5. Confusion matrix



```
3 / 3 misclassifications of 40 Magnolia_Salicifolia as 24 Cornus_Macrophylla
2 / 3 misclassifications of 20 Celtis_Koraiensis as 95 Ulmus_Bergmanniana
2 / 3 misclassifications of 72 Quercus_Phellos as 88 Rhododendron_x_Russellianum
2 / 3 misclassifications of 88 Rhododendron_x_Russellianum as 72 Quercus_Phellos
2 / 3 misclassifications of 30 Eucalyptus_Urnigera as 57 Quercus_Chrysolepis
```

figure 6. diagnostic output – suggests edge texture focused augmentation might help

K-Nearest Neighbor classifier

As a check of the relative quality of the CNN classifier, I performed a comparison classification using scikitlearn K Nearest Neighbor classifier and using the pre-computed feature data.

Nearest Neighbor Result.

90% accuracy

Conclusion

Quality of Results

Considering the inherent disadvantages of the image-based CNN for this application, it performed relatively well and without any use of domain knowledge.

I believe the CNN classifier was at a disadvantage compared to the feature-based classifier due to the images that were used. The pre-computed data contains richer features, that are the result of advanced image processing techniques – whether by image processing or manual taxonomy is not clear. Particularly, the "interior texture histogram" information is missing in the case of silhouette images – which are devoid of texture and contain only white space.

I believe the accuracy can be improved by the following:

- more augmentation
    - need to use generator approach due to massive memory slowdown of my method
- additional training images
- a more structured approach to CNN architecture tuning.

I ran out of time to implement these improvements.

NOTE:

When I originally created the CNN model, and augmented the image data, some "Leakage" occurred in the test/train split – causing artificially high accuracy, ~88% - because some of the augmented images were added to the test partition. When I discovered and remedied the problem, the CNN accuracy dropped to ~72-75%, and I didn't have time to re-architect it.