Fernando Trias
CSCI E-89 Deep Learning, Spring 2018
Prof. Zoran B. Djordjević

Topic: Identify Tree Species from Leaf Image

Problem Statement

For broad-leaf trees, it is often possible for an expert to identify the species of tree by merely observing a leaf. However, it is difficult for non-experts to do so. Many computer systems use image processing to identify properties of the leaves, like branching patterns, points and shape, and then traverse a decision tree, but this method is unreliable. I wanted to investigate if a deep neural network could identify the species of a tree by only looking at an image of a leaf.

Overview of Technology

I used Keras with Tensorflow to test various models, including MobileNet and a simpler convolutional network. The model was run on Amazon AWS p2.xlarge instances.

Overview of Steps

Define problem statement
Obtain data
Exploratory analysis
Preprocess data
Create various models
Evaluate results and choose best model

Dataset

Leafsnap is a collaboration by researchers from Columbia University, the University of Maryland, and the Smithsonian Institution. They provide an App that identifies trees from an image of a leaf by identifying features and then traversing a decision tree. In addition, they provide 976 MB of leaf images http://leafsnap.com/static/dataset/leafsnap-dataset.tar. I used a subset covering 159 Northeastern broadleaf trees.

Lessons Learned

MobileNet performed surprising well in identifying a leaf with a test accuracy of 0.90. The correct tree was identified in the top 3 predictions with an accuracy of 0.99. These results surpassed other attempts I review in the literature, but more research is required. Also, a proper field test should be employed, as well as increasing the training and testing size.

Data Overview

I looked at various databases of leaves. After several hours I encountered Leafsnap, which is a collaboration by researchers from Columbia University, the University of Maryland, and the Smithsonian Institution. They created an App for Android and iOS that identifies trees from an image of a leaf by identifying features and then traversing a decision tree. In addition, they provide 976 MB of leaf images http://leafsnap.com/static/dataset/leafsnap-dataset.tar. I used a subset covering 159 broadleaf trees common in the Northeast.

I removed Pine, Cedar, Spruce, Fir and other non-broadleaf trees because those trees cannot be identified by pictures of the leaves or needles. Some are identified by size, which is not possible to determine from a picture. Others by the 3D qualities of the needles (round, triangular, square), which is also not shown in a picture.

After designing and starting my experiments, I discovered a promising new dataset at: http://www.imageclef.org/lifeclef/2017/plant. However, I chose to continue with Leafsnap because of the high quality of images and my limited time.

Leaf Anatomy

A person would identify a tree based on many properties, including the leaf, bark, shape, flowers, location, etc. To identify solely based on the leaf, a person would look at the overall shape of the leaf and individual features such as the vein pattern, smoothness or roughness of the surface, ridges along the edge or surface and sometimes color.

Veins

As with animals, trees have a cardiovascular system. The leaf is composed of veins that transport water and minerals. The patterns can be very small. However, usually the larger branching pattern on the leaf is sufficient.

Color

Although the color of a leaf can be used to distinguish the species, because of lighting and other issues, the colors in the photos were not reliable enough to make this determination. As a consequence, I converted all photos to black and white, which helped to speed up training.

Size

Leaves come in all shapes and sizes. The proportion of width to height is important in identifying a leaf. A convolutional network would not be able to pick that up on a macro scale, but the proportions at a smaller scale are also important. A convolutional layer should be able to pick that up.

In ImageNet, images are typically resized without regard to the width-height proportions. If I were to do that, a leaf such as the one on the left below would be resized to appear as the leaf on the right.

However, this may not work well because the leaf on the right may look like a different species. As a result, I used a resizing method that preserved the width-height proportions in the hopes that this would improve reliability.

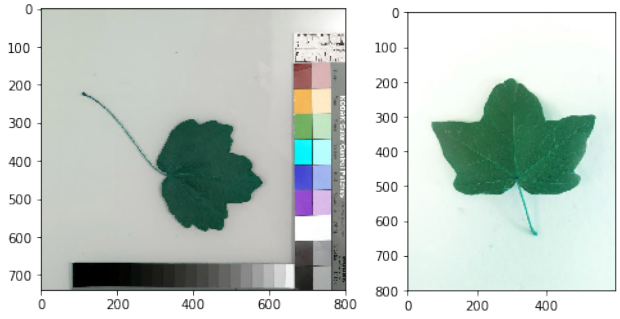
When scaling the photos, I chose a resolution of 224x224, because I initially planned to use a ResNet network (although I finally used MobileNet). The resolution should be small enough to make training practical, but it need to be high enough to show the veining pattern of the leaves and any serrations on the leaf edges. 224x224 was the absolute minimum resolution where the veins were still distinguishable.

Deep Networks

I wanted to try two different networks to test our reliability and speed. First, I wanted to try as simpler CNN as a benchmark. Initially, I considered using ResNet or Inception. However, after reading about MobileNet and Tensorflow Lite, I thought that using it would allow me to create an App that could be used by amateurs to identify trees in the field. Thus, I chose to test MobileNet.

Datasets

The Leafsnap dataset has two types of images: Field and Lab. Field images show a leaf on a white background. Lab images include a color reference bar. For illustration, here are two images of the Acer campestre (Field maple):



When resizing images, I try to crop the Lab images to remove the color bars. In theory, the color bars can be used to calibrate the color of the images in order to use that to identify the leaf, but that task is complex and so I just remove them.

Note as well that the leaves have different orientations. During training, I used rotation to augment the data in the hopes that this will enable the network to identify leaves in any orientation.

I broke the dataset into 3 parts: Training, Validation and Testing. There are 159 trees represented. Each dataset had to contain at least one image from each tree class. The datasets did

not contain any overlapping or common images. I used 80% for training, 10% for validation and 10% for testing.

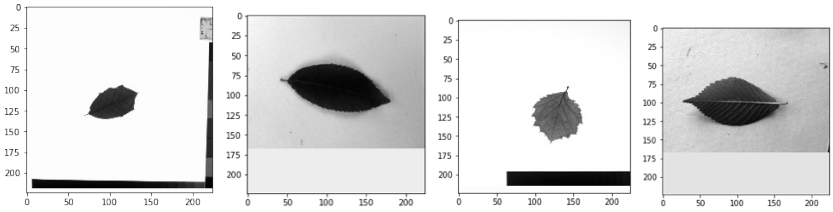
	Images	Percent
Training	21,285	80%
Validation	2,661	10%
Testing	2,657	10%

The Jupyter Notebook leaf-setup.ipynb was used to create these data sets following this strategy:

1. Unzip Leafsnap dataset
2. Import index
3. Exclude non-broadleaf trees (pine, cedar, etc).
4. For each tree, randomly assign sample images to training, validation, testing with desired probabilities.
5. Convert image to black-and-white
6. Crop and resize preserving aspect ratio
7. Save image to appropriate directory

The actual training is performed by leaf-run.py. It will set up the model, run the optimizer and then test the results.

After processing, typical images look like this:



Augmentation

Because the source data is limited and because leaves can come in any orientation, I use data augmentation. Specifically, I use Keras's built-in augmentation provided by ImageDataGenerator():

1. Rotation range: 180 degrees
2. Width shift: 0.2
3. Height shift: 0.2
4. Shear: 0.2
5. Zoom: 0.2

CNN

The CNN I created was composed of 3 Convolutional layers, a Flattening layer, a Dense layer and another Dense layer for output. The Keras summary is:

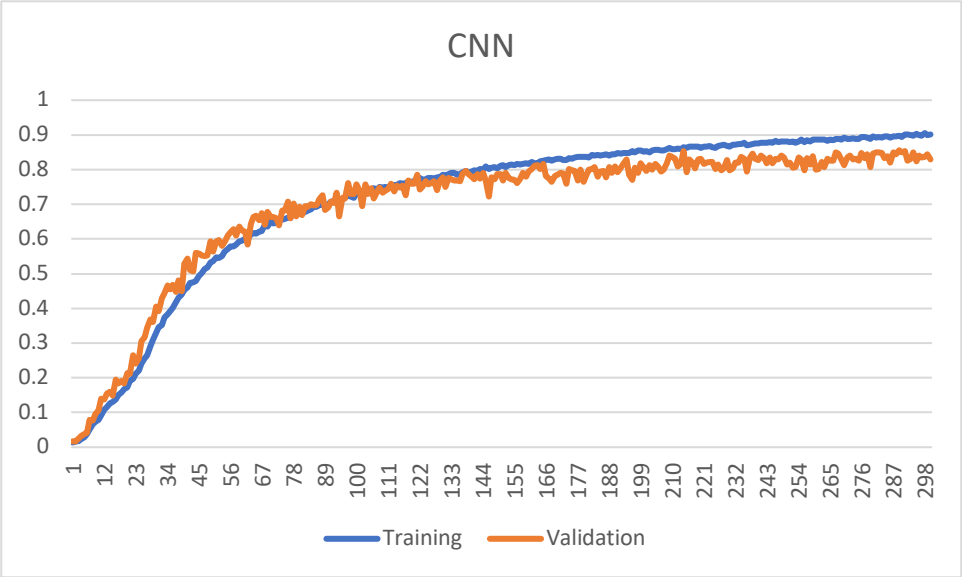
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 64)      640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0
```

```
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 128)     73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 56, 56, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 200704)            0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                12845120
_________________________________________________________________
dense_2 (Dense)              (None, 159)               10335
=================================================================
Total params: 13,225,119
Trainable params: 13,225,119
Non-trainable params: 0
```

I added Relu regularization to each layer in order to avoid overfitting, which I expected would be a problem since the leaves are all very similar.

I used the SGD optimizer because I had read that it is more reliable. After a few smaller tests, I chose a learning rate of 0.01 and 300 epochs because the smaller tests were slow to make progress. However, after reviewing the results shown below in the chart, it seems that less epochs would have yielded similar results.

Trainable parameters	13,225,119
Epochs	300
Training accuracy	0.9016
Validation accuracy	0.8290
Test accuracy	0.8438
Training time	16.47 hours

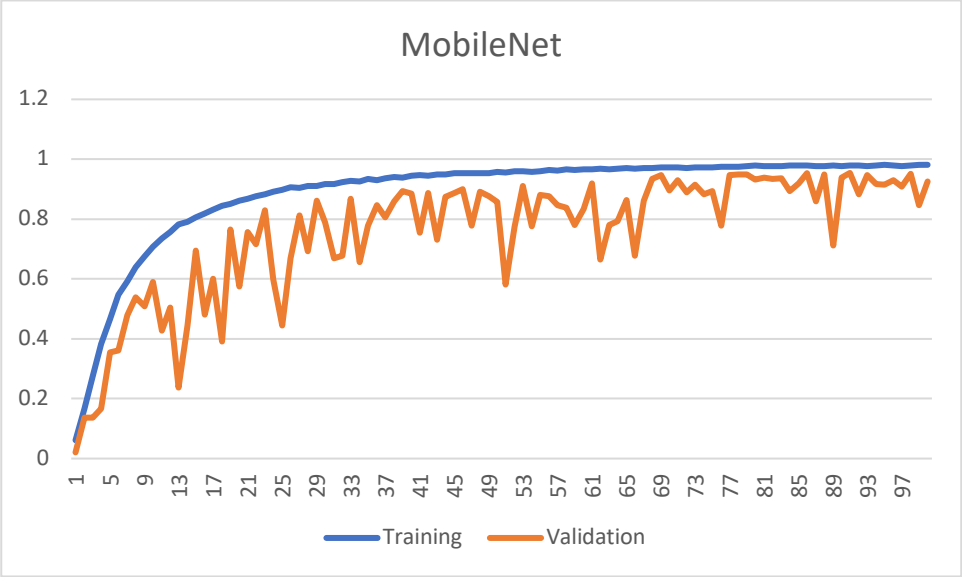


MobileNet

The MobileNet network is much more complex than the CNN so I will now show the Keras summary. Instead, this is a more compact summary from the web:

Like in the CNN model above, I used SGD with a learning rate of 0.01. Training on this network was much faster than the CNN model. Thus, instead of 300 epochs, I ran only 100. But looking at the chart below, even 100 is excessive. Instead of 16.5 hours of training for the CNN, the MobileNet model took only 2.5 hours to train to a superior accuracy. However, the validation curve was a more jagged, perhaps indicating that training was not as reliably consistent.

Trainable parameters	3,369,375
Epochs	100
Training accuracy	0.9814
Validation accuracy	0.9248
Test accuracy	0.90625
Training time	2.52 hours



Other Networks

In addition to CNN and MobileNet, early in my exploration I also did trials of ResNet50 and Inception. Both the trials were promising, achieving some level of accuracy after a 20 epoch test. However, due to limited time I decided to focus on CNN and MobileNet with the eventual goal of making a mobile app.

Comparison of CNN and MobileNet

MobileNet was superior to CNN in every way, which is not surprising. However, I was a bit surprised that CNN took so much longer to train, given that it is simpler in many ways. It has almost 4 times as many parameters, but yet took over twice as long to run to 100 epochs (and over 6 times longer to run to 300 epochs, while still not matching the accuracy of MobileNet).

	MobileNet	CNN
Trainable parameters	3,369,375	13,225,119
Epochs	100	300
Training accuracy	0.9814	0.9016
Validation accuracy	0.9248	0.8290
Test accuracy	0.90625	0.8438
Training time	2.52 hours	16.47 hours

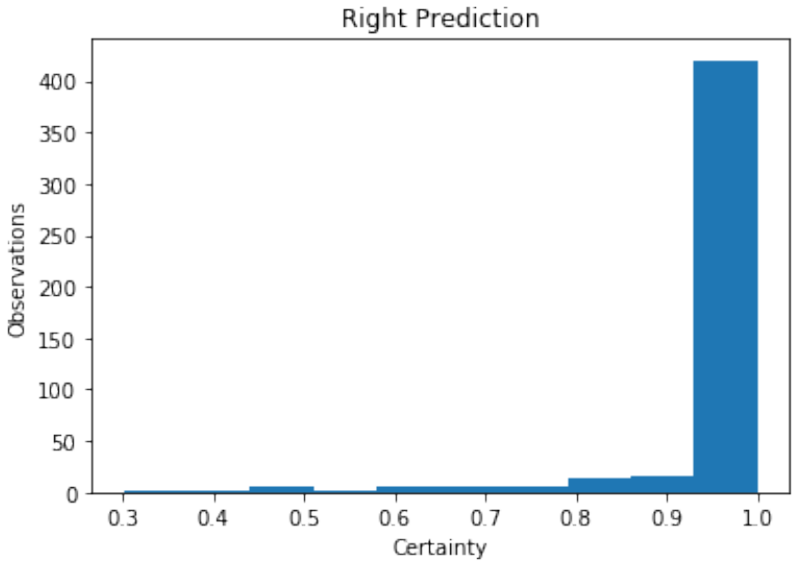
Certainty of MobileNet Model

To evaluate the reliability of the models, I created a Jupyter Notebook called test.ipynb which preforms simple tests on the models. One of the tests is showing if the model correctly predicts the image in the top N predictions. I ran this over 500 testing data images for the top 1 to 5 predictions.

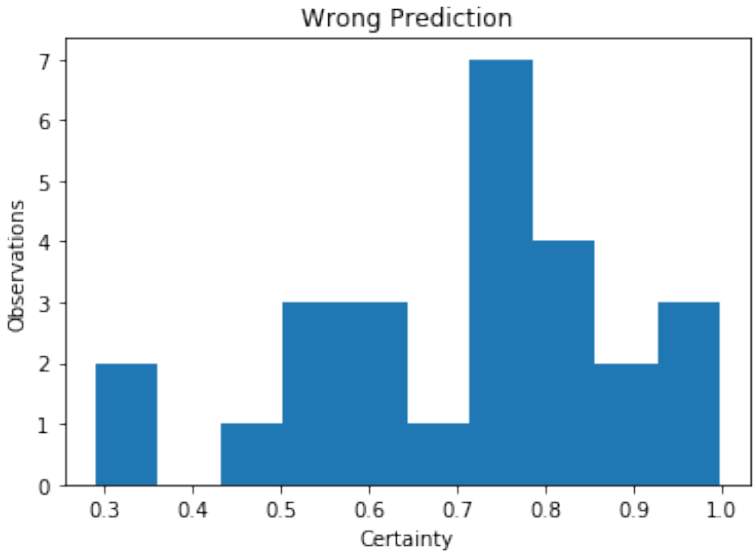
Top N	Accuracy
1	0.92
2	0.98
3	0.99
4	0.99
5	0.99

In addition, I looked at the distribution of probabilities in the resulting prediction vector. Because each prediction is a number from 0 to 1, it is conceptually similar to a probability. I evaluated 500 test images and split the results into two groups based on whether the top prediction was right or wrong. Then I looked at a histogram of the distributions of the probability of the top prediction.

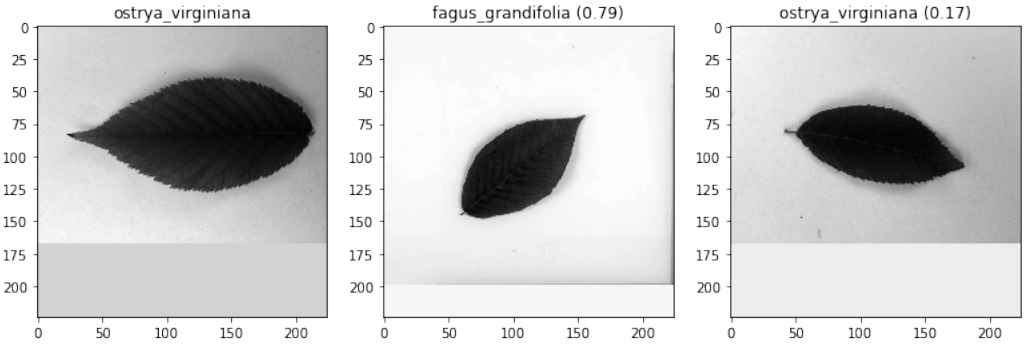
For predictions that were right, as expected the probability of the top prediction is generally high.



For predictions that were wrong, the probability is more mixed.



This is probably because some trees have leaves that look identical in a photograph. Distinguishing features might be the color or texture or perhaps vein patterns that are very small. In addition, some trees with similar leaves have very different shape, color and bark. For example, here is one incorrect prediction with corresponding probability numbers.



Fagus grandifolia (American Beech) and Ostrya virginiana (American hophornbeam) have virtually identical leaf images. But trees have very different size, shape and bark.

Here is Ostrya virginiana (American hophornbeam)

And here is Fagus grandifolia (American Beech)

Thus, a more accurate system would also look at images of at least the bark and possibly the overall shape.

Tensorflow Lite and CoreML

My plan was to convert the MobileNet model to run in Tensorflow Lite. I attempted this but was unable to do so due to various incompatabilities with the version of Keras and Tensorflow I used for training and the version required by Tensorflow Lite. After several hours devoted to this without success, I decide to investigate other options.

Apple has a CoreML package that is similar in goals to Tensorflow Lite. I was able to easily convert my model's H5 file to the Apple's format using Apple's simple tool. I then dragged the model into Xcode and was able to work with the data. Unfortunately, I have not yet gotten camera access to work and thus haven't produced a working App.

Youtube Videos

Two minute (short): https://youtu.be/S6jOuE9KHlg
15 minutes (long): https://youtu.be/TKbMa8bZzLQ

References

"Leafsnap: A Computer Vision System for Automatic Plant Species Identification," N Kumar, P Belhumeur, A Biswas, D Jacobs, W Kress, I Lopez, J Soares, Proceedings of the 12th European Conference on Computer Vision (ECCV), October 2012.

"Plant leaf recognition using shape features and colour histogram with k-nearest neighbor classifier," T Munisami, M Ramsurn, S Kishnah, S Pudaruth, Second International Symposium on Computer Vision and the the Internet, 2015.

"Leaves recognition system using a neural network," B Sekeroglu, Y Inan, 12th International Conference on Application of Fuzzy Systems and Soft Computing, August 2016.

"Deep Residual Learning for Image Recognition," Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015.

"A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network," S. Wu, et al., July 2007.

LifeCLEF 2017 Lab Overview: multimedia species identification challenges," A. Joly, et al., CLEF, 2017. http://www.imageclef.org/lifeclef/2017/plant

What Tree Is That? Arbor Day Foundation, April 2009.