Classifying Tree Species with Convolutional Neural Networks

Branden Strochinsky

CS 5137 - Machine Learning

December 8 2017

Introduction/Motivation

For my project I tried to predict the species of a leaf from an image of it. I did this using the Leafsnap database (kumar, et al., October 2012) and a convolutional neural network. The motivation for this project started from an interest in computer vision/image classification. Convolutional neural networks are a type of neural network that excel at image classification, so learning how they work is important for an understanding of computer vision. Leafs were chosen because the task seemed challenging, but identifying leafs by looking at them is a task that humans with domain knowledge can accomplish. That means that there are visual features that distinguish species that a computer could learn. The challenge of this problem was to get a computer to be able to recognize these features well enough to be able to accurately predict the species.

Basic Approach

As stated previously, my approach to this problem was to use a convolutional neural network or a CNN. My CNN takes an image of a leaf as an input and out puts a vector predicting the class/species. A CNN is similar to other neural networks in that there is an input layer, an output layer, and hidden layers in-between. The hidden layers of a CNN consist of convolutional layers, pooling layers, and fully connected layers. In the convolutional layers a matrix of weights, called a filter, is scanned over the width and height of the image, and the dot product of the filter and that section of input is computed. After that, an activation function is applied to the results of the dot products, and that is the output of the convolution step. This can be seen in figure 1 that I created.

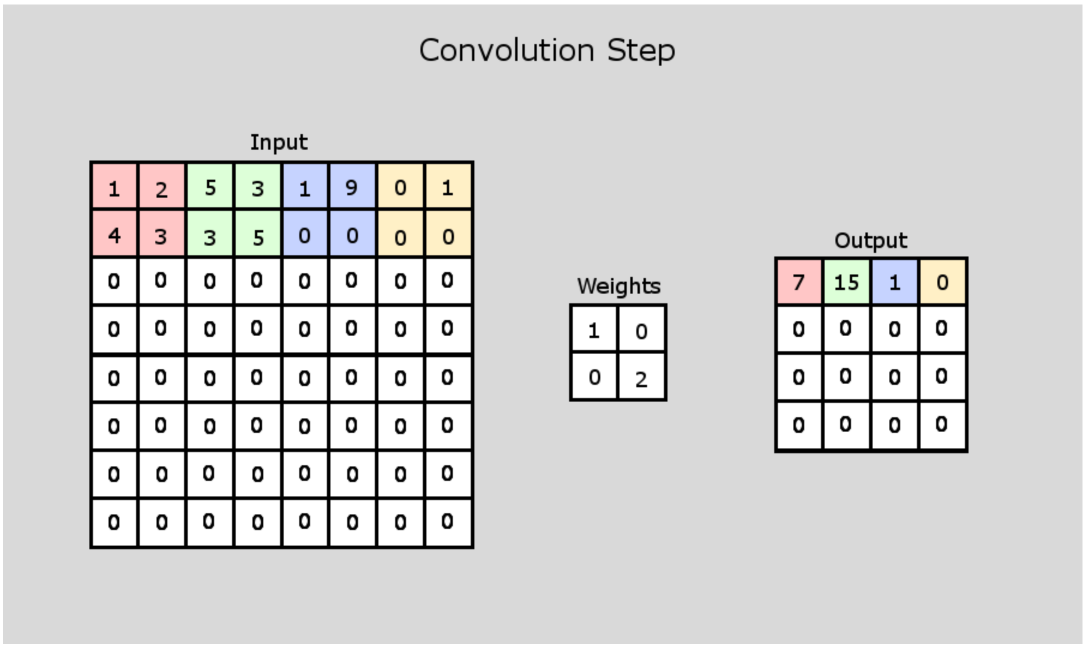

Figure 1 - Convolution with 2x2 filter and stride 2

Convolution layers are typically followed by pooling layers. In pooling layers clusters of neurons from the outputs from the convolution layer are combined in to a single neuron. For example, max pooling takes the maximum value from the previous layer as the output. This can been seen in figure 2 that I created.

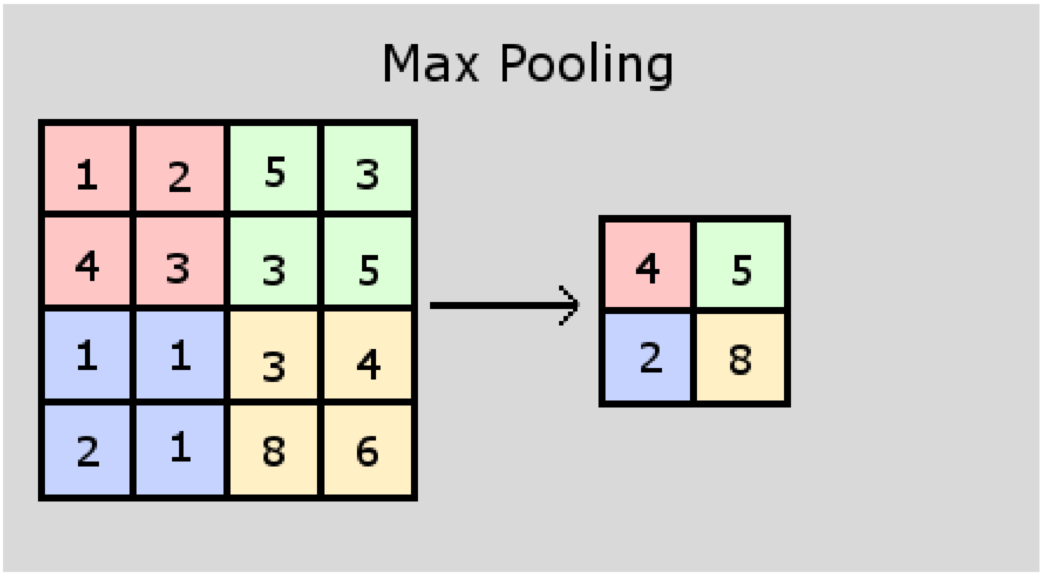

Figure 2-Max pooling with a 2x2 filter and stride 2

The final layer of a CNN is the fully connected layer. This layers takes every neuron from the previous layers as the input and output a vector of length n, where n is the number of classes. This final vector is used to predict the class. The weights of the CNN are determined by training it like any other neural network. The output is compared to the actual class for that input and back propagation is used to update the weights.

My CNN is based off the one in *Tree Species Identification Based on Convolutional Neural Networks* (Zhou, Yan, & Huang, 2016). It contains two convolution layers, two pooling layers and one fully connect layer. This basic structure can be seen in figure 3. There are 12 filters in the first convolutional layer and 24 in the second. The convolution kernel is 3x3. These were chosen to mirror the model in "*Tree Species Identification Based on Convolutional Neural Networks*. After each convolution layer my CNN applies the activation function ReLu to the output. Relu is defined as:

f(x) = max(0,x)

This function is faster to apply than other activation functions like sigmoid.

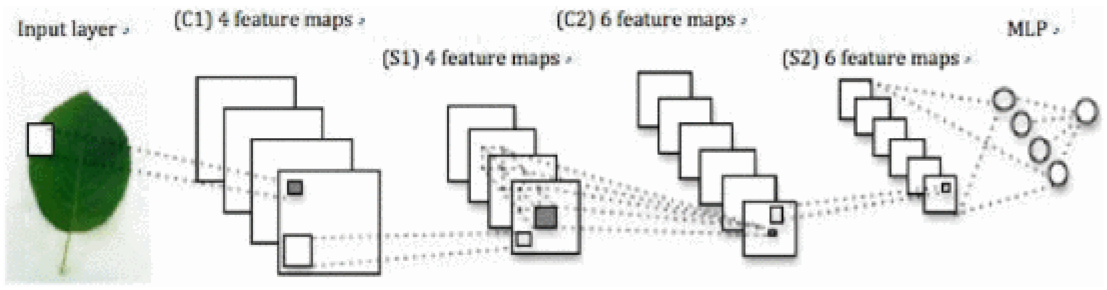

Figure 3

Experimental Setup

The Leafsnap dataset contains 30,866 images of leafs that cover 185 tree species. Only ten out of the 185 species were used in the experiment. This was to lower the scope of the project and make it more computationaly feasable. The species with the most images were used to maximize the amount of training data. The amount of training data was increased by mirroring some of the original images and including those.

The Leafsnap dataset contains automatically-generated segmentations for each image, these are images that show the separation of the leaf and the background of the image. The first step in using the leaf images was to set all the background pixels to white. This was done by referencing the segmentations to determine which pixels were background pixels and setting all 3 color channels of that pixel to the max value of 255. Next each image was scaled down to be 50x50 using bilinear filtering. This entire process can be seen in figure 4. After selecting only the images from the 10 species, and doing the mirroring, I ended up with a total of 2,401 images. Twenty percent of this images were randomly chosen for testing while the reaming 80% were used to train the CNN.

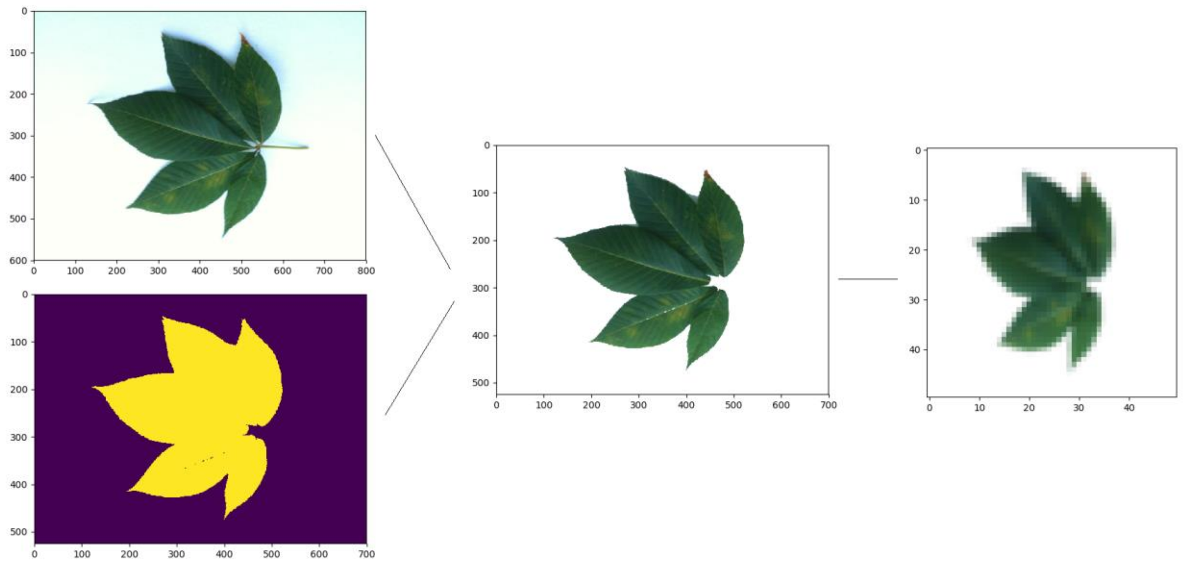

Figure 4

Experimental Results

The CNN was trained with the proposed model and images of the 10 tree species. A learning rate of 0.01 was used. After training the model 200 times, the model was tested against the images that were not used to train it. It was able to predict the species of tree of these images with 65% accuracy. While it was training its error rate for each epoch was also recorded and can be seen in figure 5.

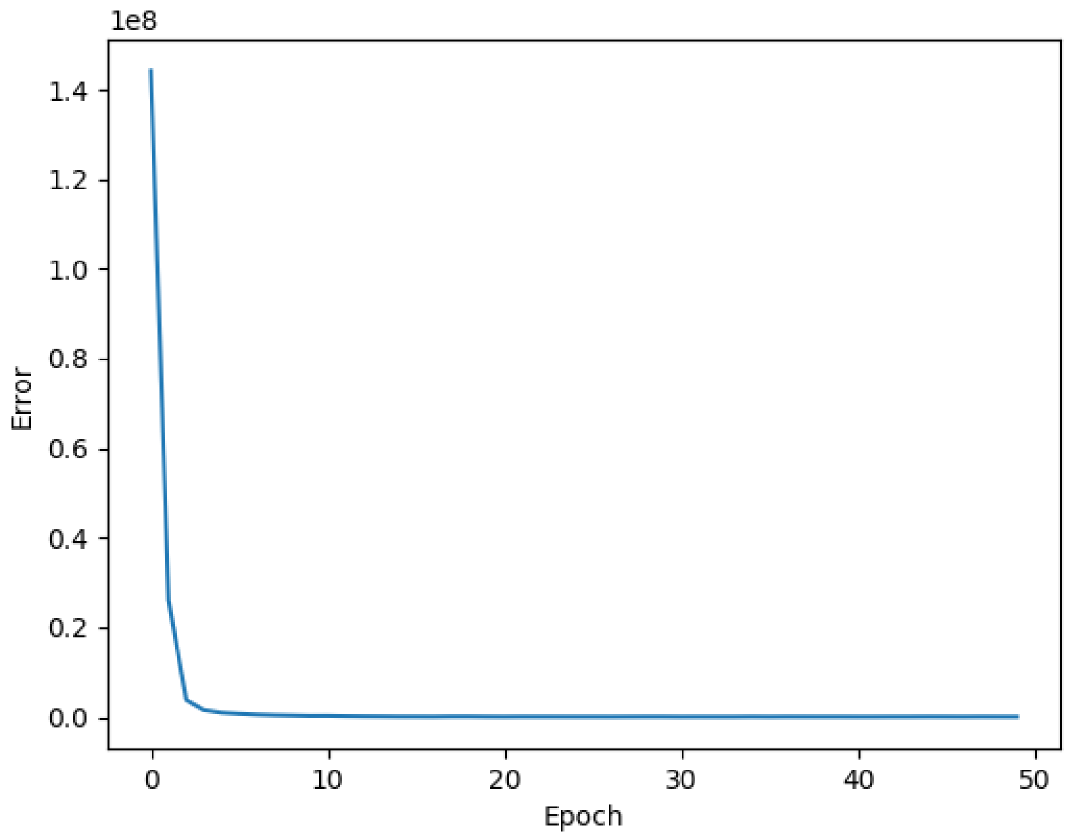

Figure 5

After training and testing it once I doubled the number of convolution filters at each layer in the CNN and retrained it. This did not seem to effect the accuracy of the model, so I did not try making any further adjustments to it.

Conclusion

Overall I believe that the accuracy showed that a convolutional neural network can be used to accurately predict the tree species from a leaf image. In H. Zhou, C. Yan and H. Huang's paper they were able to get the accuracy up to 90%. This is significantly higher than my 65%. I believe part of the issue is the orientation of the images. They manually oriented every image so that the leaves were pointing in the same direction. I believe this would have helped the accuracy of my results but it would take a lot of time to manually orient all of the images by myself. Further, I believe that with more training data I would be able to improve the accuracy. The LeafSnap dataset has a lot more images than what I used. A lot of these images were not useable for my project without manually cropping them to isolate the leafs. If would probably be worth it to spend the time cropping and orienting these images if a higher accuracy was needed.

Another possible avenue to explore would be the hyper parameters. These are the parameters in the model that are static and not learned. These include the size of the convolution filters, the number of convolution layers/pooling layers, and many other parameters. My initial values were chosen from H. Zhou, C. Yan and H. Huang’s paper. Although I spent some time changing them, there are a lot of values I did not try that could have a significant effect on the results.

References

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

Zhou, H., Yan, C., & Huang, H. (2016). Tree Species Identification Based on Convolutional Neural Networks. *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics.* Hangzhou.