Justin Tsao

Capstone Project 2 Milestone Report – Classifying Poisonous plant with CNNs

Proposal

This project aims to solve identifying the leaves of poisonous plants using images of poison ivy, poison oak, and poison sumac. The idea came from the fact that I found out I was deathly allergic to poison ivy during a walk in the woods.

Many people who are allergic to poisonous plants can benefit from an image classification model for poisonous plants. Regardless of the severity of allergic reactions, being able to avoid these plants without prior knowledge as to how to identify them from sight can save a lot of trouble, and in my case a trip to the hospital. There can also be the potential for business clients as well. Different outdoor companies who sell hiking or outdoor sports products could use the image classification in the form of a mobile app to help promote their company and products. Perhaps, the poisonous plant classification can be a feature in an outdoor hiking app which provides users with other various information on the outdoors like fishing spots, edible plants, mushrooms to avoid etc. It could possibly use in a suite of mobile apps for image classification on poisonous mushrooms, poisonous plants, edible plants, and so on. The other client can be national park systems. They could have an app that hikers or park-goers can download on their smart phones.

For the data I have initially been scraping 100s of pictures of poison ivy, oak, and sumac from google images. I have been checking the images removing bad images and making sure the plants are the correct plants. Other sources of data I have been looking into are image databases.

I will attempt to solve this problem by first compiling a list of images and labeling them as the correct poisonous plants. Then, I will analyze the images and format them into workable resolutions and pixel sizes to be used in a neural network. Next, I plan to use Keras/Tensorflow to develop and train a convolutional neural network. The whole process will follow a typical machine learning flow of splitting the data into training and test sets, tuning model parameters, and in this case the adding or removing of different types of layers and nodes in the neural network.

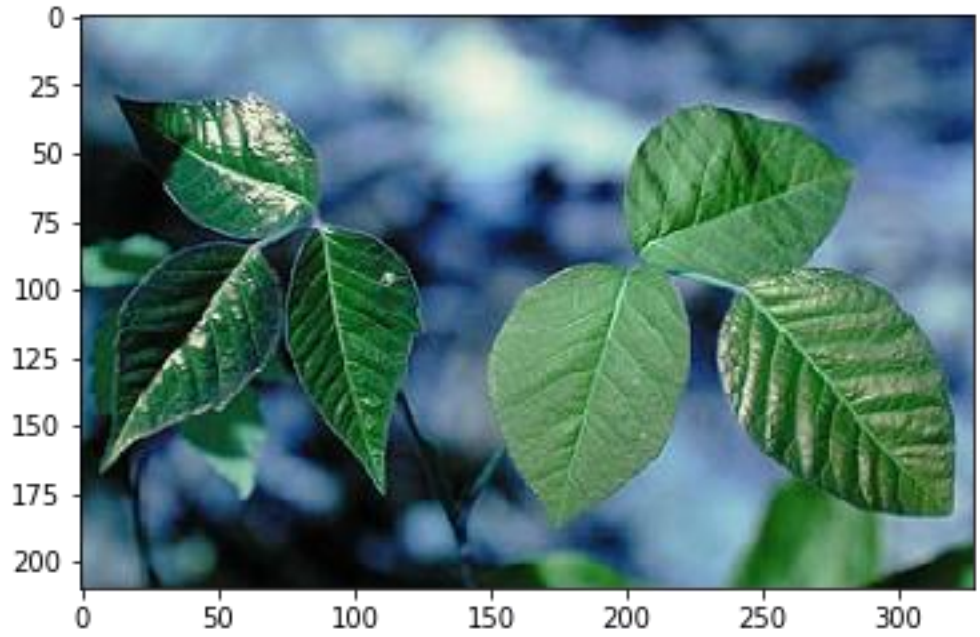
The deliverables for this project will be various Jupyter Notebooks, written reports, and slide presentation. The possibility to create a web application may present itself given the time constraints of the project.

Data

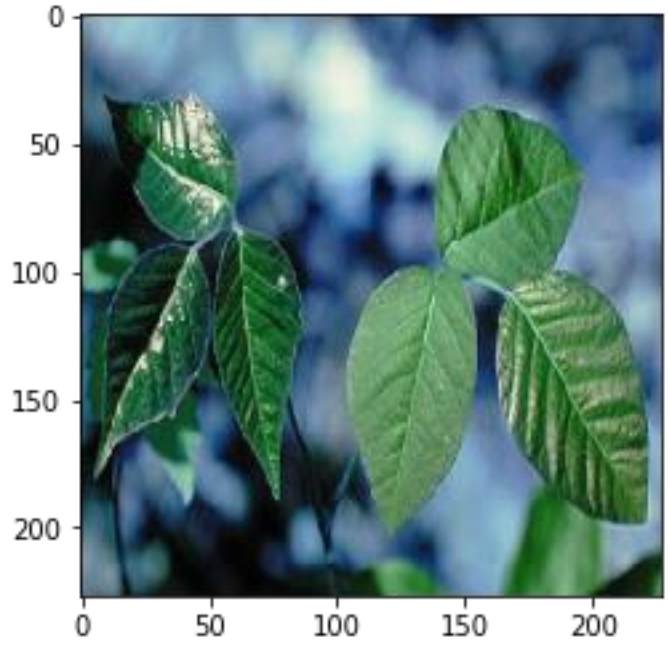
For this project we required pictures of 3 different types of plants. These included pictures of poison ivy, poison sumac, and poison oak. Based on search results from the internet, these three plants seemed like the most common. To obtain lots of images in a reasonable amount of time, I used a Python module called *google-images-download* which can be found here https://github.com/hardikvasa/google-images-download. To use this library and download more than 100 images at a time, one must download *chromedriver* which can be found here http://chromedriver.chromium.org/. Running this module, I attempted to download at least 500 images from *Google Images* using the key words ‘poison ivy leaves’, ‘poison oak leaves’, and ‘poison sumac leaves’. These images will automatically be downloaded into separate directories where the directory names will be used as labels that will be fed into a neural network.

I was unable to find at least 500 of each image. Plenty of pictures of poison ivy and poison oak were found, but it was difficult to find pictures of poison sumac. I also attempted to find images from other databases and websites as well. Once I was done searching, I also had to look through each image individually to weed out any pictures that were not good fits for the project. Many of these images had to be removed to non-related images or images that had text, or other objects obscuring the actual image of the plants. Also, there were many misclassified images which had to be moved to the correct directories. Though, I am no expert in poisonous plants, I do have some experience as a child growing up in the country side of North Carolina and was a boy scout. I was confident in my ability to recognize the correct images. What I ended up with was 334 (46%) images of poison ivy, 287 (40%) images of poison oak, and 102 (14%) images of poison sumac. I was a little disappoint in the lack of poison sumac images. What I learned from a project like this, is that gathering a data for a project like this is both difficult and time consuming, compared to another project I did where I could 100s of thousands of data samples using an API. I was fine with this, as I wanted the full experience of performing a real world data project that where I was not hand fed clean data in large amounts.

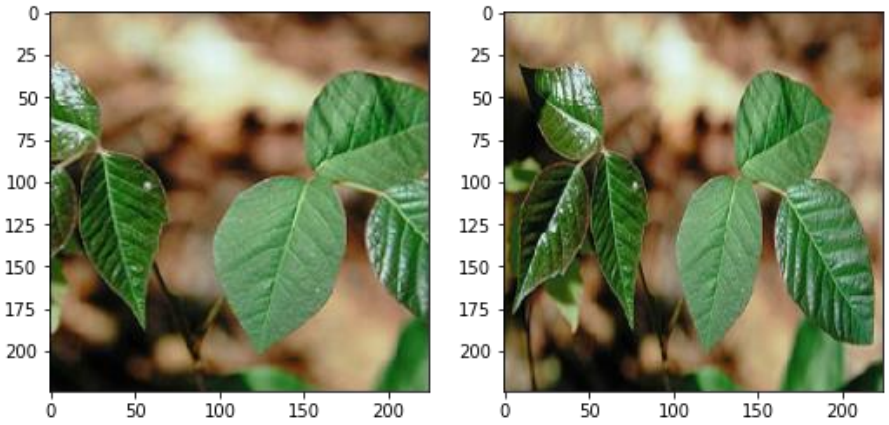
I used OpenCV and matplotlib to view a few of the images.



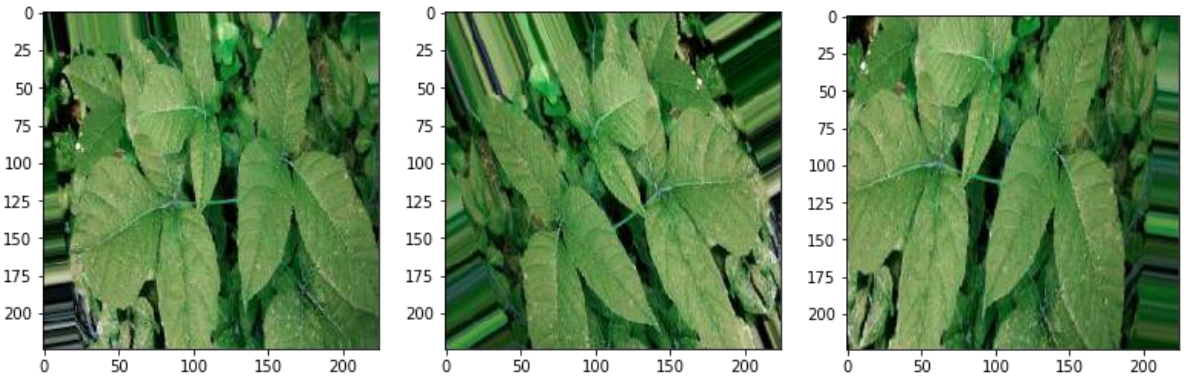
The shape of this image is (210, 328, 3) The last dimension is the number of channels, 3 in this case. This represents the RGB channels. Depending on the network I would have to resize the images to certain sizes. For example, the AlexNet convolutional network requires images of size (227x227):



In additional to resizing I also normalized the pixel values which could have a min value of 0 and max value of 255. I used *Numpy* to create an array of the image divided by 255.0. As you can see by resizing the image, the shape is distorted. I also used the same image set and decided to crop the images in the center without distorting the shape:



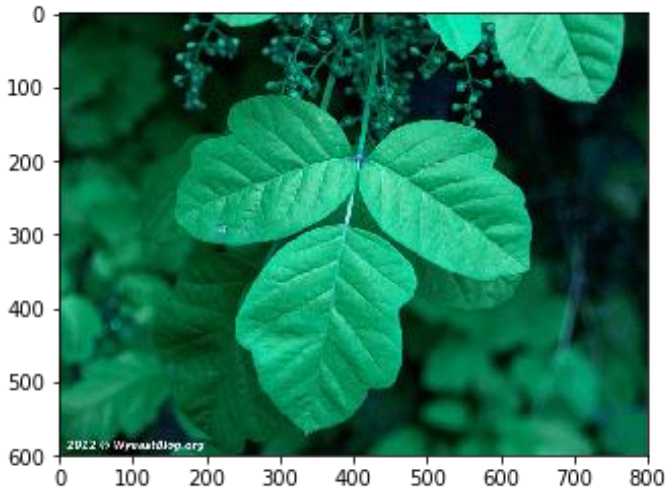
For this project I used Keras to build the layers of the CNNs and train the models. Keras has an *ImageGenerator* class which generates additional sample data that have slight distortions. I decided to use the ImageGenerator to add different rotations, slightly shift the width and height, and flips the image on its horizontal axis. ImageGenerator also adds more robustness to the data.



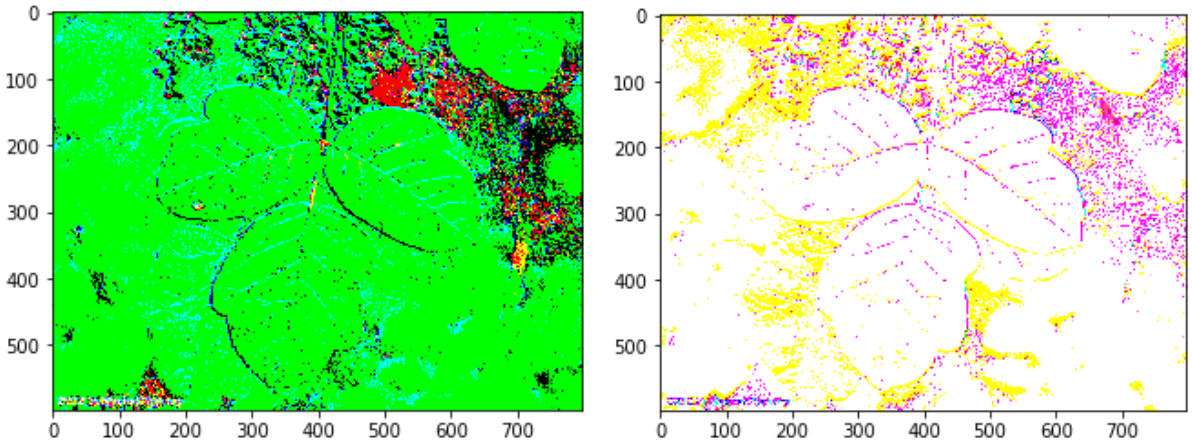
After adding the augmented images, I had 637 (38%) images of poison ivy, 596 (36%) poison oak, and 417 (26%) poison sumac.

Exploring Data and Layers of a CNN

Since I the project utilizes convolutional networks, I decided to get a better understanding of the different layers and how to computer views the images in the network. Here is a regular image.

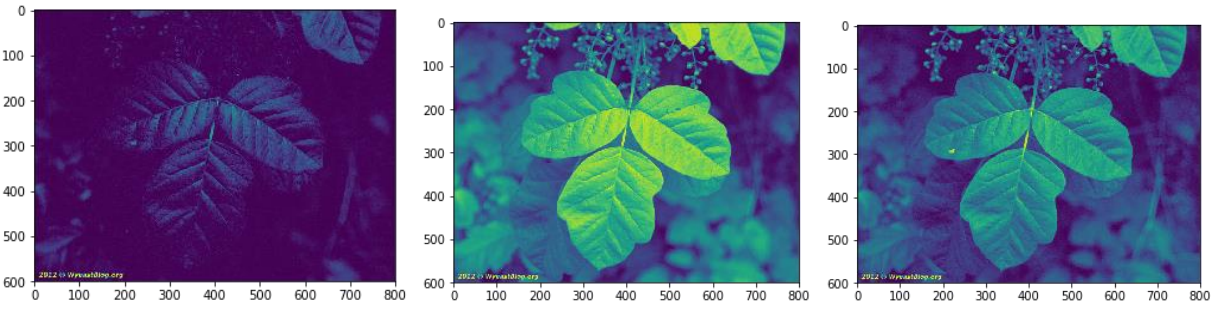


Let's look at a convolutional layer. A convolutional layer basically uses a filter called a kernel to convolve or move over the image to try and discern different features. There can be many kernels in a convolutional layer. We can not visualize more than 3 kernels at a time, so below are two versions of the image below passed through a convolutional layer each with different kernels. Each kernel is of size 3x3.

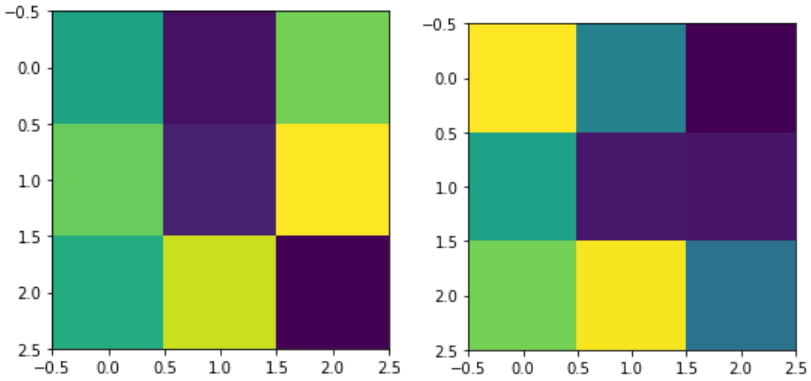


The kernels are randomly initialized when the CNN is initialized, so the result is different. The output above shows the result of the 3 channels in the RGB images.

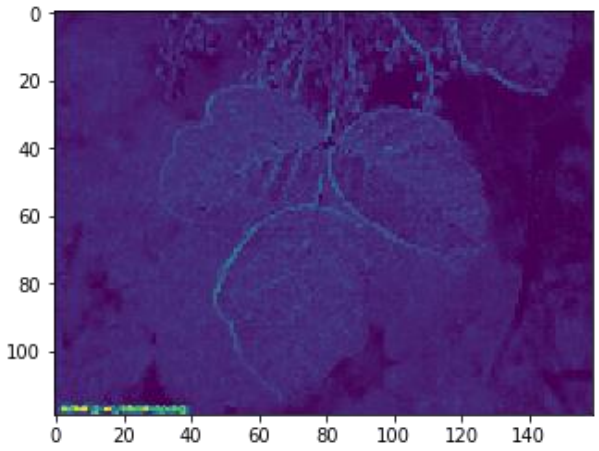
I also decided to split the image up into the 3 separate channels and pull out different kernels. Below are the three different channels.



Example of two 3x3 different kernels.



The other two common layers are pooling and activation layers. Activation layers decide if a neuron will fire or not. Pooling layers are shrunk down versions of the filtered image. A filter is walked across the image similar to the convolutional layer, except the filter will find the max value of pixel within the filter window as output. Below is the image passed through a pooling and activation layer.



As you can see, it is more blocky.

If you are curious, here is the same picture passed through the AlexNet CNN with one kenel.