Identifying Plant Species by Leaf Venation Patterns Using Machine Learning

By

Evan Parduhn

Honors Project Thesis Submitted in Partial Fulfillment

Of the Requirements for a Baccalaureate Degree “With Distinction”

In the Ron and Laura Strain Honors College of

THE UNIVERSITY OF INDIANAPOLIS

April 18th, 2020

Faculty Advisor: Paul Talaga, Ph.D.

Executive Director of Honors: James B. Williams, Ph.D

Abstract

Invasive species can be devastating to native vegetation and can be difficult to identify unless properly trained. The best method for combating invasive species is to quickly identify and contain them. This project looks at a possible solution to difficult identification by creating a trained artificial neural network using TensorFlow that can categorize plant species by their venation patterns. This network could then be integrated into an application that identifies if a particular plant was invasive and allows for faster containment measures to be taken. For this project an artificial neural network was set up and tested using five different plant species; *Acer campestre, Acer ginnala, Acer griseum, Acer platanoides, Acer negundo.* Some of these species are invasive, but this project focused on categorizing species rather than if they are invasive or native. The images used for this project were pulled from the open-source Leafsnap dataset [1]. Leafsnap is a similar project that uses machine learning to categorize leaf species by leaf shape. The network was trained on a total of 135 images, 27 for each species, and tested with 35 images, 7 images for each species. After training, the network had an overall accuracy of 94% when categorizing by venation pattern. The preliminary results of this project show that with a larger dataset and a more refined artificial neural network, a reliable application can be created that will quickly identify invasive plants.

Table of Contents

List of Tables

List of Figures

Introduction

Statement of Purpose

"While some invasive plants are distinctive and easily recognized, many others are difficult to distinguish from one or more species of our native flora"[2]. That was from the book *Mistaken Identity*, a book about invasive plants that look similar to native ones. As more and more invasive species pop up and compete with local plants, it becomes imperative to find methods of dealing with these plants. Rejmánek states that there are three objectives to dealing with invasive species; prevention, early detection, and containment/eradication [3]. Although plant recognition may not be as helpful in preventing an invasion of a non-native species, it could become imperative in identifying and eradicating existing invasive plants. Rejmánek states that to achieve the second objective, early detection, "proper field experience and relevant sampling techniques (e.g. adaptive sampling, Thompson & Seber 1996) are necessary" [3].

Artificial neural networks are the leading technology in artificial intelligence and are used in many technologies such as facial recognition and self-driving. Facial recognition works by processing an image at low resolution to reject non-face regions and then evaluates the challenging regions at higher and higher resolutions [4]. The purpose of this project was to use neural networks to identify plant species by their venation patterns. It could be used to categorize invasive species and as a method for early identification and help prevent the damage caused by invasive plants.

State of the Art & Related work

In this section I will discuss three things; how leaf identification is being done or how it has been done, describe one existing automated leaf identification method, and provide a background of machine learning.

Current State of the Art

There are many resources that people can rely upon to identify plants, but there is always a need for the user to either be trained in identification or have an extensive amount of time to dedicate to identification. For example there are many books and websites that catalog species with information on the plant. Go Botany[5], for example, asks you several questions about the plant and returns a list matching those descriptors. The site only covers so many features before the user has to begin comparing one by one, which can still end up being tedious and time consuming. Another issue with this site is that it categorizes plants by location, which could cut out species that might not normally exist in that area such as invasive species [5].

A method of finding invasive plants and determining impact in the area is aerial hyperspectral photos. Hyperspectral imaging works by collecting information from the electromagnetic spectrum. Certain objects have 'spectral fingerprints', which can be used to identify a certain material. After an aerial hyperspectral image is taken, someone can map where and how widespread invasive species are [6].This method of identification is crucial for early detection and containment. This method is not efficient because the cost for taking

hyperspectral images is high and an expert that can differentiate species is required to create the mappings.

The methods currently being used are not ideal for curtailing invasive species because they are either too slow or require extensive training. A possible solution to this problem is to create an application that categorizes plant species automatically from cell phone images.

Leafsnap – An Automated Identification Tool

One such technology for plant identification that is already widely available is Leafsnap [1]. A leaf recognition app that was developed using facial recognition software and artificial intelligence. Kumar states why this software could help save time:

"*Without visual recognition tools such as Leafsnap, a dichotomous key (decision tree) must be manually navigated to search the many branches and seemingly endless nodes of the taxonomic tree. Identifying a single species using this process – by answering dozens of often-ambiguous questions, such as, "are the leaves flat and thin?" – may take several minutes or even hours. This is difficult for experts, and exceedingly so (or even impossible) for amateurs.*" [1].

Manual leaf identification requires extensive training and experience before someone can be reliably trusted to identify a species. An artificial neural network, on the other hand, would help speed up the identification of invasive plants and also make it easier for amateurs to find invasive plants.

Leafsnap works by finding the outline of the leaf from an image, extracting curvature based features, and then identifies the leaf using a nearest neighbor search algorithm[1]. The nearest neighbor algorithm starts at one point in a map of comparable data, where each node is

connected by similar features to another node. Data that is extracted from the leaf is then compared to the nodes connected to the starting point and moves the node with the closest match. This repeats until a threshold for a certain match is reached. Leafsnap then returns the top 25 matches and boasts that 96.8% of queries have a match in the top five results. Leafsnap is able to discern 189 different species, so they are able to reduce the possible species by 98% and have a 96.8% accuracy. Our work used a much smaller number of species, so our performance will be based on our top predicted species rather than top 5.

Machine Learning

Leafsnap solves the problem of identification by implementing an algorithm that compares existing data on features to an unidentified leaf. Another possible method for solving this problem is using an artificial neural network. Essentially, a machine is given a number of Tasks (T), which it is then rated on by its performance (P). The machine is supposed to improve with experience (E) [7]. For example, if we wanted to train a machine to differentiate between plant leaves, the machine's tasks would be to take a picture and identify it. The machine would then be rated by whether it got the identification right or wrong. If the machine gets it right, it is rewarded making that outcome more likely. If the machine is incorrect, the connections leading to that outcome are inhibited and make the outcome less likely. In comparison, Leafsnap uses a nearest neighbor approach, which is equivalent to a neural network with no hidden layers which can only separate species if their feature space is linearly separable. Neural networks with hidden(middle) layers are able to discern a feature space with more complex boundaries.

In a little more detail, we understand that the machine is a neural network, where each node takes multiple inputs, and has one output. So by having many nodes at the beginning and slowly decreasing the number of nodes, we can filter a lot of binary input into one simple output. Each of these nodes has weights, which really means how much is needed to flip the nodes switch and change the output [8]. This is shown in Fig 1. Each of the circles is a node, and each of the arrows is a connection to another node. Each of these connections has a weight to it that determines where the information gets sent next until it reaches the end. The node in the output layer that has the most activation is the answer.

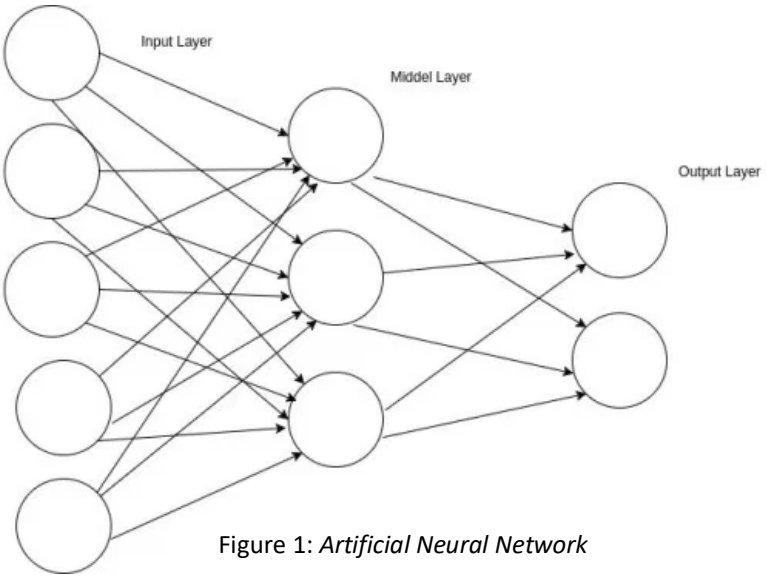

Figure 1: *Artificial Neural Network*

For this project the number of nodes at the beginning is 256. This will then boil down to five nodes at the end, matching the number of species this network was being trained on. Data is sent to each node, weighted to decide what node it will send this data to next. This continues until one of the final nodes is reached, which is the machine's final answer. When training, if the network outputs a wrong answer, there is a feedback mechanism that weakens the weights of the nodes associated with that incorrect answer. This makes that specific wrong answer less likely to happen. The same feedback mechanism occurs when the network outputs a right

answer. This time, the feedback mechanism strengthens those paths making the answer more likely to happen. After many training samples the network should learn and improve its performance.

Justification and Honors Worthiness

Invasive species have been a plight in the world for as long as people have been travelling. These species take away from native plants resources, leaving native plants starved and dying. From the U.S. Forest Service, “Invasive species have contributed to the decline of 42% of U. S. endangered and threatened species” [9]. The only way to prevent this from happening is early detection and eradication. One issue with early detection is that many people are not trained to differentiate plants and some invasive plants look similar to native plants. This issue could be ameliorated by using a neural network, which is easier and faster to train than humans, to identify these plants.

I chose to create a system that identified plant species by their venation patterns. Based on my research, I found that most currently available applications identify leaves based on the outline of the leaf. Through this research I will determine if plants can be identified by their venation patterns, and if this method of identification could be used and incorporated with existing methods to bolster accuracy.

Solution

This project had two goals. The first goal was processing the Leafsnap dataset so that the leaf is isolated from the background. The second goal was to create the neural network and train it to identify plant species using the vein patterns in the leaf.

Dataset

Figure 2: *Acer Campestre* Lab Image

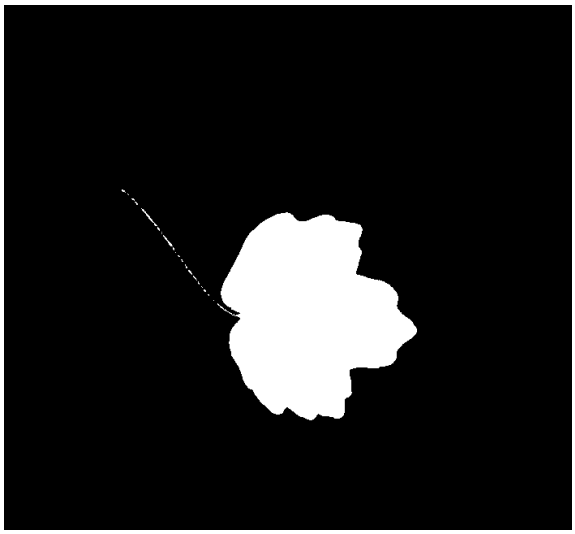

Figure 3: *Acer Campestre* Segmented Image

The network I created needed to categorize leaves by their venation patterns. As it is, the Leafsnap dataset is not conducive to this type of categorization and needs to be changed to fit those needs. The Leafsnap dataset contains the original picture taken in a lab setting and a segmented image of that picture that Leafsnap uses in their current system. Figure 2, above, is an example of an original lab image. This image is not ideal for training a network because there is a lot of superfluous data such as the color bars on the bottom and right side. Since the network needs to categorize by vein patterns, this extra data might interfere with the categorization process and produce a network that identifies the leaves by the color bar instead of the vein pattern. Figure 3, above, is an example of a segmented image. Figure 2 was processed by Leafsnap's application and turned into Figure 3. This allowed Leafsnap to identify the species by the shape of the leaf. Figure 3 is ideal for Leafsnap's system, but not for this system because it got rid of the venation patterns in Figure 2. This project needed an image that

preserved the contents of the leaf but got rid of any arbitrary data around the leaf. To accomplish this, the segmented images were used as masks for the original image.

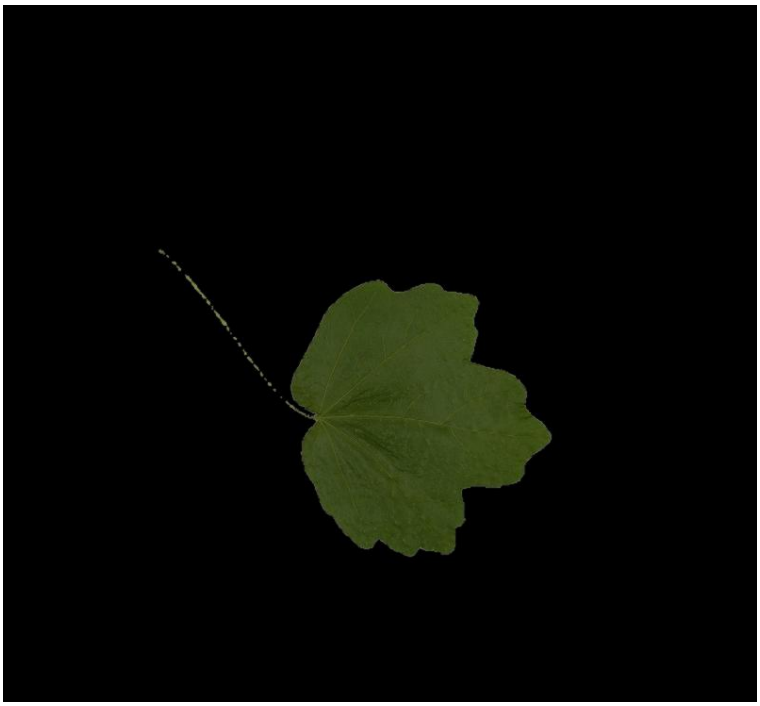

Figure 4: *Acer Campestre* Masked Image

Figure 4, above, is an example of what we have done to the dataset to make it ideal for the network I created. Figure 4 was created using a software called GIMP, an open source image editing software. GIMP has a feature called Python-Fu which allows for a script that uses GIMP functions to run on multiple images. The script I wrote takes the original image and the segmented image and puts them on top of each other, resizing when necessary. Then the script creates a grayscale mask of the segmented image and inverts it. This cuts out the white space

and leaves a cutout of the leaf. Then the original image and the mask are combined to create something similar to Figure 4. This script was run for all of the images in the Leafsnap dataset.

Once the dataset was processed to meet the needs of the network, a script was developed that created a neural network, generated a histogram of the images, and trained the network using those histograms, described below.

Neural Network

There were two types of data that were used as input to the neural network; a histogram of the image, and a ratio of the area of veins to the total area of the leaf. A histogram of the image gets the color distribution of the image. Since the veins of the leaf are a much different color than the rest of the leaf, a histogram will show the concentration of veins to the rest of the leaf. The ratio contains the same information, but it is more focused. We needed to keep both because the ratio does not contain all of the veins and the histogram is not as focused on vein patterns.

The histogram was generated by flattening an image array into a list of numbers. The ratio was generated by finding the area of the veins and dividing that by the total area of the leaf. To find the area of the edges, an image was first sent to Canny, a function from the library cv2. Canny detects edges by finding a large difference in color and then creates a black and white image of just the edges.

These two pieces of data were chosen because they both contain a lot of focused information. The neural network is trying to categorize the information that is given to it and it decides what is useful or not within the information. For example, if I try to train a neural

network to identify whether something in an image is either a dog or a dolphin, the neural network might use information in the background of the image, rather than the animal. Most pictures of dolphins are going to have a blue background because they live in the ocean. Pictures of dogs, on the other hand, will most likely not have a lot of blue in the background. If this data is not filtered out, the neural network might decide that all images with a blue background have dolphins in them. That is why it is important to choose data that is relevant to what the neural network should be trained on. A histogram of the image preserves the color distribution and contains information of just the leaf on vein concentration. The ratio contains that concentration as well and has the bonus of being impervious to changes in the zoom of the image.

Discussion

Results

Five different species were arbitrarily chosen to train and test the network. In total, there were 135 samples in the training set, 27 per species, and 35 samples in the testing set, 7 per species. The network achieved a 94% accuracy with its first choice. Since any given network can be saved and used again, there is no need to discuss average accuracy across different networks. Instead we will discuss the network that achieved the highest accuracy.

Acer campestre	*Acer ginnala*	*Acer griseum*	*Acer platanoides*	*Acer negundo*
100%	71%	100%	100%	100%

Table 1: Accuracy by Species

Table 1 shows the accuracy that the neural network makes the correct match. *Acer ginnala* is the only leaf that the network does not categorize correctly. The network sometimes categorizes an *Acer ginnala* leaf as either *Acer platanoides* or *Acer negundo.* There is no discernible reason for this error.

Reflection

This project set up a dataset that can be used for future machine learning applications and created a neural network that can identify three species at an 94% success rate. The original goal for this project was to use machine learning to create an application that identifies leaves as either invasive or non-invasive. This project is a large step in the direction of this goal since it sets up an artificial neural network that can be trained to identify leaves. The missing link is a larger dataset that includes leaves in different orientations and ones that are taken in the field.

The reason that I was not able to get to the application phase of this project was largely due to the issues faced while creating the dataset. The original goals were created with the assumption that I would be able to use the Leafsnap dataset right out of the box. However, the Leafsnap dataset was not formatted in a way that I could easily set up a neural network. This meant that I had to spend much more time analyzing the dataset and creating a new dataset that allowed for training based on vein patterns. Through this issue, though, I have learned a lot about image processing and using GIMP scripts as a tool to process a lot of images quickly and efficiently. This is a valuable skill for anyone to have and could be useful for me specifically as I enter the professional world where this skill would set me apart.

I have also learned a lot about machine learning, and specifically about how to use Tensorflow, which is a machine learning tool. These skills are valuable because there are many

problems in the world that could be solved with machine learning. Now that I have the skillset to use this tool, I am better prepared to face professional problems that machine learning can solve.

Future Work

Neural Network

Currently the maximum accuracy that the neural network can reach is 94% with five species. This is comparable to Leafsnap as they claim that they have a 96% accuracy within the first five results across 189 different species. The next steps for this project on the machine learning side will be to train a network with more species and ensure that an accuracy rating of 94% or higher is maintained.

It might be beneficial to explore different types of data for the training set which would raise the accuracy. This would include finding more information about the veins, such as the angles between them, their length, and their curve. This data could be difficult to find as lines are harder to map across pixels, but it would be very valuable and could raise the accuracy.

Dataset

The biggest next step for this project would be obtaining a more comprehensive dataset that includes other types of images. Currently the data set I have created includes only lab images. These images ensured the leaves were flat in varying degrees of light which allow for the best data to be extracted. This network would fail on images that were taken in the field

that are either curved, have an indistinguishable background, or do not contain the contrast necessary to retrieve useful data.

Conclusion

Invasive species have a destructive impact on native species that can cause plant species to become endangered. The methods that are currently being used to contain invasive plants rely on early detection. Without extensive training, properly identifying invasive species is not fast enough. This project is a step towards solving the problem of not being able to identify invasive species quickly or accurately. This project uses machine learning to train an artificial neural network to categorize five species by their venation patterns. The network achieved an average accuracy rating of 94% which is 2% lower than the Leafsnap project. To complete this project, a larger dataset is needed that includes more image types and more invasive species. Once a more comprehensive dataset has been compiled, an application can be made that could be used by anyone to identify invasive species and help stop their plight.

Literature Cited

[1] "Leafsnap: A Computer Vision System for Automatic Plant Species Identification," Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida C. Lopez, João V. B. Soares, Proceedings of the 12th European Conference on Computer Vision (ECCV), October 2012.

[2]Sarver, Matthew, et al. *Mistaken Identity?: Invasive Plants and Their Native Look-Alikes: an Identification Guide for the Mid-Atlantic*. Delaware Department Agriculture, 2008.

[3] Rejmánek, Marcel. "Invasive plants: approaches and predictions." Austral ecology 25.5 (2000): 497-506.

[4] Shah, Jay. “Neural Networks for Beginners: Popular Types and Applications.” *Stats and Bots*, Stats and Bots, 16 Nov. 2017, blog.statsbot.co/neural-networks-for-beginners-d99f2235efca.

[5] Native Plant Trust, 14 April 2020, https://gobotany.nativeplanttrust.org/simple/woody-plants/woody-angiosperms/#_filters=family,genus,habitat_general,state_distribution,plant_habit_wa,leaf_type_general_wa,leaf_arrangement_wa,leaf_blade_margin_general_wa,leaf_duration_wa,armature_wa&habitat_general=terrestrial&plant_habit_wa=tree&leaf_type_general_wa=simple&leaf_arrangement_wa=1%20leaf%20per%20node&leaf_blade_margin_general_wa=toothed&leaf_duration_wa=deciduous%20or%20marcescent&armature_wa=absent&_view=photos&_show=plant%20form

[6] Lawrence, Rick L., Shana D. Wood, and Roger L. Sheley. "Mapping invasive plants using hyperspectral imagery and Breiman Cutler classifications (RandomForest)." *Remote Sensing of Environment* 100.3 (2006): 356-362.

[7] Goodfellow, Ian, Yoshua Bengio and Aaron Courville. *Deep Learning*. Cambridge, Massachusetts, MIT Press, 2016. http://www.deeplearningbook.org

[8] Nielsen, Michael A. Neural networks and deep learning. Vol. 25. USA: Determination press, 2015.

[9] "Invasive Plants." *Forest Service Shield*, U.S. Forest Service, www.fs.fed.us/wildflowers/invasives/.

Appendix

Appendix A: Acknowledgements

I would like to acknowledge Dr. Paul Talaga for his help on this project. As my advisor he has been a great resource for troubleshooting or just to discuss new ideas. This project would not have been possible without him.

Appendix B: leaf_id.py

```
# -*- coding: utf-8 -*-
"""leaf_id.ipynb
Author: Evan Parduhn
Description: This script is used to train an artificial neural network to categorize
        leaves according to their venation patterns
Automatically generated by Colaboratory.

Original file is located at
   https://colab.research.google.com/drive/1goioTPOEeN8p4QkvPZLDrbbJsnrn_BoS
"""
```

```
from __future__ import absolute_import, division, print_function, unicode_literals
# Install TensorFlow
import tensorflow as tf
from google.colab import drive
import os
import math
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
import PIL
# This needs to be catered to the users Google drive. the ls only serve as a manual check
drive.mount('/content/drive')
!ls "/content/drive/Shared drives/"
!ls "/content/drive/Shared drives/EvanLeafSnap/"
!ls "/content/drive/Shared drives/EvanLeafSnap/leafsnap-dataset"
!ls "/content/drive/Shared drives/EvanLeafSnap/Vein_Identifier"

# This function takes a filename and parses out the extra characters and returns just the name
# of the species. This was made specifically for the data set I created so it may need to be changed
def parseName(filename):
  fl = filename[:-5]
  if(fl[-2].isdigit()):
    fl = fl[:-2]
  elif(fl[-1].isdigit()):
    fl = fl[:-1]
  return fl
fp = "/content/drive/Shared drives/EvanLeafSnap/Vein_Identifier/Test"
iname = ""
im_dict = {}   # Keeps track of the number samples for each species found
tr_dict = {}   # Contains a list of the samples for each species in the training list
te_dict = {}   # Contains a list of the samples for each species in the testing list
dy = {}        # This is the answer key
i=0

for filename in os.listdir(fp):
  fl = parseName(filename)
```

```
 if(fl not in im_dict.keys()):
  im_dict[fl] = 1
  tr_dict[fl] = []
  te_dict[fl] = []
  dy[fl] = i
  i+=1
 elif(fl in im_dict.keys()):
  im_dict[fl] += 1
 # This find the minimum number of samples to a species
 # I did this so that the number of samples across both lists would remain the samples
 # to hopefull prevent scewed results
 i_min = min(im_dict.values())
 train_num = int(i_min * .8)
 test_num = int(i_min * .2)

# print(train_num,test_num) Used to debug
# print(dy)

# Finds the histogram of and image and returns it
def genHist(fn):
 im = PIL.Image.open(fn)
 im = np.array(im)
 hist,bins = np.histogram(im,bins=255, range=(0,255))
 return hist

# Finds the edges in an image and creates a new image with just the edges
# Returns the sum of the new image which is the area of the edges
def genEdges(fn):
 img = cv2.imread(fn)
 gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
 edges = cv2.Canny(gray,20,50,apertureSize = 3)
 return np.sum(np.asarray(edges))

# Uses genEdges to find the area of veins and returns that over the area of the leaf
def genRatio(fn):
 im = PIL.Image.open(fn)
 im = np.array(im)
 area = 0
```

```
 for arr in im:
  for p in arr:
   if(p.any() != 0):
    area += 1
 return int(genEdges(fn) / area)

curr_name = ""
# x arrays contain samples, y arrays contain the names
x_train = []
x_test = []
y_train = []
y_test = []

# Evenly distributes samples into the training and testing arrays
for filename in os.listdir(fp):
 fl = parseName(filename)
 if(len(tr_dict[fl]) < train_num):
  x_train.append(np.append(genHist(fp + '/' + filename),genRatio(fp + '/' + filename)))
  y_train.append(dy[fl])
  tr_dict[fl].append(genHist(fp + '/' + filename))
 elif(len(tr_dict[fl]) >= train_num and len(te_dict[fl]) <= test_num):
  x_test.append(np.append(genHist(fp + '/' + filename),genRatio(fp + '/' + filename)))
  y_test.append(dy[fl])
  te_dict[fl].append(filename)
# You have to convert these to numpy arrays so they work with TensorFlow
# Yes I know that numpy has an append function, but its confusing and I hate it
x_train = np.asarray(x_train)
x_test = np.asarray(x_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)
# print(len(x_train),len(x_test),len(y_train),len(y_test)) # for debugging

# These are useful functions if you want to send whole images to TensorFlow
# I did not do that for the final implementation, but they are here for future implementations

# Finds and returns the max shape in the training and testing arrays
def findMaxShape(arr_tr, arr_ts):
 ms = [0,0]
```

```
 for im in arr_tr:
  if(im.shape[0] > ms[0]):
   ms[0] = im.shape[0]
  if(im.shape[1] > ms[1]):
   ms[1] = im.shape[1]
 for im in arr_ts:
  if(im.shape[0] > ms[0]):
   ms[0] = im.shape[0]
  if(im.shape[1] > ms[1]):
   ms[1] = im.shape[1]
 return ms

# adds padding to the samples according to the max shape returned by findMaxShape
# Funny how the function names say exactly what they do
# This is necessary because TensorFlow likes its training samples to all be the same size
def addPadding(arr_tr, arr_ts):
 ms = findMaxShape(arr_tr,arr_ts)
 n_tr = []
 n_ts = []
 for im in arr_tr:
  hd = ms[0] - im.shape[0]
  wd = ms[1] - im.shape[1]
  nim = np.pad(im,((0,hd),(0,wd)),'minimum')
  n_tr.append(nim)
 for im in arr_ts:
  hd = ms[0] - im.shape[0]
  wd = ms[1] - im.shape[1]
  nim = np.pad(im,((0,hd),(0,wd)),'minimum')
  n_ts.append(nim)
 return np.asarray(n_tr),np.asarray(n_ts)
# print(len(x_train),len(x_test),len(y_train),len(y_test)) # more debugging

# Creates the training model in TensorFlow. Trains and then tests. I also added a predict
# to see more detailed results
model = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(input_shape=(256,)),
 tf.keras.layers.Dense(128, activation='relu'),
 tf.keras.layers.Dropout(0.2),
```

```
  tf.keras.layers.Dense(len(dy), activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=200)
model.evaluate(x_test,  y_test, verbose=2)
model.predict(x_test)
```