Stevie Magaco

Jonathan Burnside

Machine Intelligence Systems

Post-mortem: Distinguish images of dogs from cats

My research project was based on the Kaggle competition: Dogs vs. Cats Redux: Kernels Edition. Most of my research was based on the image recognition tutorial on the website: ww.pythonprogramming.net. This tutorial instructs us on how to build a basic Digit recognizer which is a bit like what I had to do, which was to be able to tell the difference between 2 things (cats and dogs). Initially I thought that this would be easy, as I did not need to compare between 10 different types of images. It did not take long for me to realize that I was very mistaken, as I had not considered the format of the images I’d be taking in. For the digits, the images where simple and contained nothing more than the digit itself, on the other hand, my images contained much more, such as people, other animals (like dogs in cat images and cats in dog images), scenery, etc. To sum it up, every sample I was given to train with, contained very specific detail on every single pixel that I needed to compare. This meant that whenever I compared these pixels, there was a very high chance that I was not going to get any hits. This issue was also paired with the fact that when I was trying to build and save my pixel arrays, so that I could to check for comparison later, they would end up as an enormous file (+4Gigs). To solve this, first I had to reduce the size of the image, from 512x512(262,144 pixels per image) to 64x64(4,096 pixels per image). This alone reduced the size of the files to 2% of its original size. Next, I reduced the variable range for each variable in a pixel, the standard ranges from 0 to 255, this range caused issues when making comparisons as the range was to wide to get a match. The new range I decided to implement (from 0 to 13) allowed for much better results, as this meant that it becomes a lot easier to get a match.

There are 3 main functions in the program: CatvDogthreshold(x) = this returns array of pixels with a shorter range, CATvDogcreateExamples() = this creates our saved pixels array to be traversed and checked, CATvDogwhatNumIsThis(x) = this returns a graph of the cat or dog hits/matches and a label about describing what the image is,.

There are 2 ways to get results, one is through the CatvDogwhatNumusThis(x) function. This function returns detailed results about a single selected image, you will need to input the path of the image, I used the test samples that ranges through 1 to 12500. The second way, is through the Submission() function. This function runs through a selected set of test samples and output a csv called submission.

The one big down side for one this project is that the traversal is very slow, as it behaves very much like a breadth first traversal, another issue that also contributed to the slow down, is how the pixel array is saved locally. I used a .txt file, and separated each array by adding a tag (cat:: or dog::) to identify what that array is supposed to be. The problem here is that every time I want to parse through it, I have to break it up first (by calling the split function). This could be solved by saving the arrays in a .csv file, and instead of having to manually break the arrays I could simply call the column or row number to get that value for comparison.