Week 4:

The new code does not use dictionaries. In this program dictionaries are converted into ndarray where in one column their RAM address and the value in that RAM is present. This program is very costly as one more for loop (98 loops) are added in the program.

Now when i tried to implement njit or vectorize I got errors. I made a sample(small) program which has the same logic as we have in our program and tried to make changes and implement vectorize and njit and was able to successfully implement them and the resultant increment in speed is same.

I made a simple neural network and one matrix multiplication program and successfully implemented cuda in these programs. The resultant speed is same if I implement njit or vectorize in that.

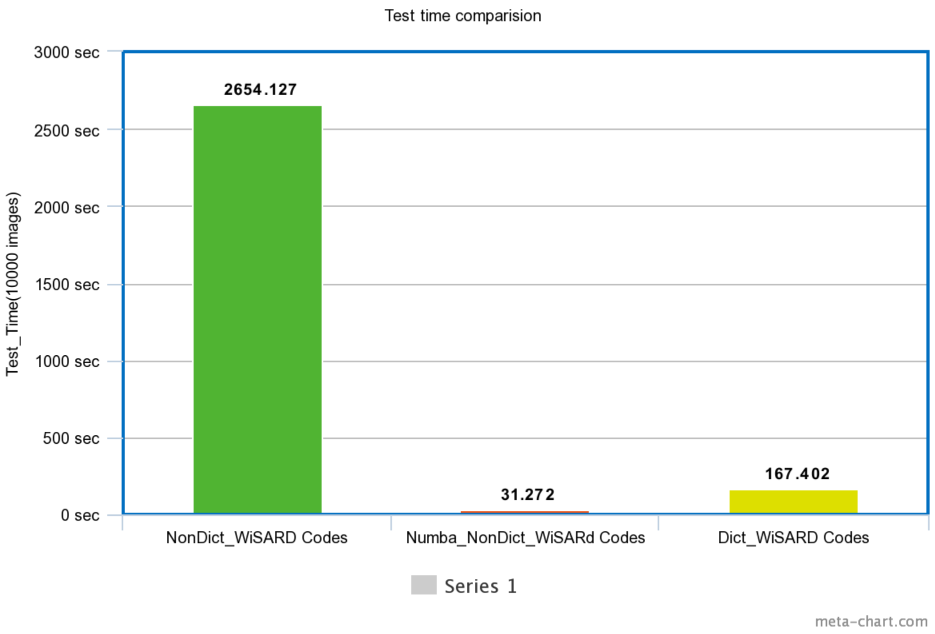
Our program has 4 for loops and the logical way to increase the speed in these is by applying njit which is made to lift the loops in the program.

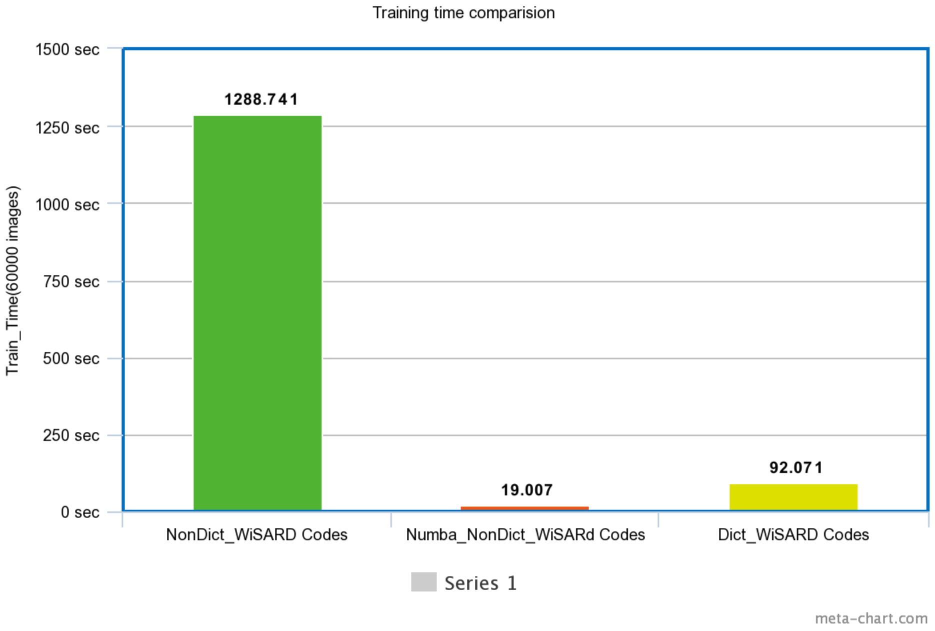
In our program there are several issues while implementing njit like we cannot use enumerate, for x in list, max, str etc. which makes python easy to use. So I converted everything like c++ programing and then implemented njit which was then successfully implemented and increased the speed of training module by 67.8 times and the testing module by 84.87 times.

Results:

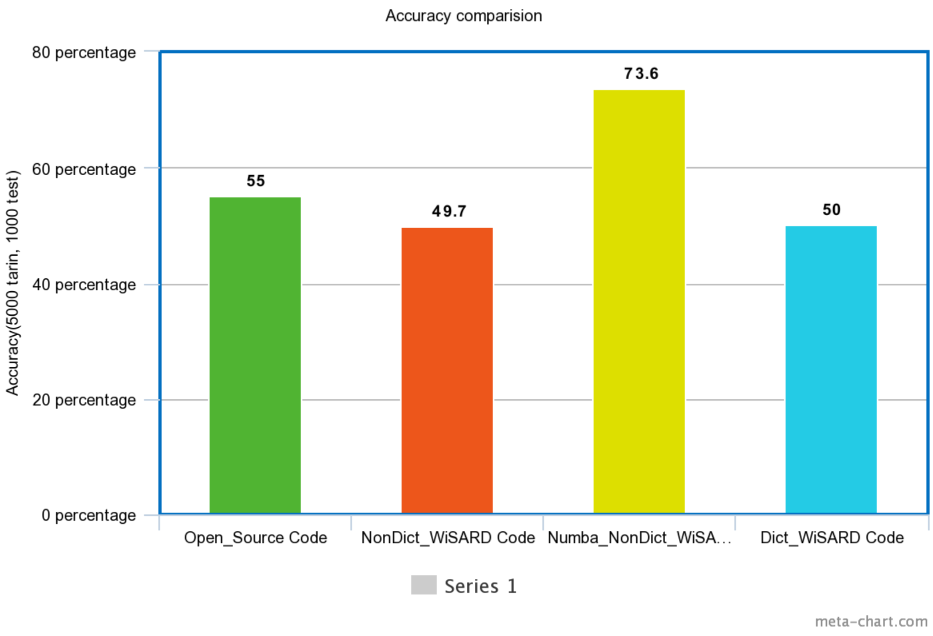
If we train the network by 5000 images without numba the time taken was 183.053 seconds, the testing time of 1000 images was 455.939 seconds. Now once we implemented numba the training time of 5000 images was 1.965 seconds, the testing time of 1000 images was 3.395. This is still 4.84 times faster to train then the initial code (with dictionaries) and 5.3 times faster to test.

The training time of 60,000 images was 1288.741 seconds, the testing time was 2654.127 seconds. Now once we implemented numba the training time was 19.007 seconds; the testing time was 31.272 seconds. This is still 6.72 times faster to train then the initial code with dictionaries and 8.27 times faster to test.

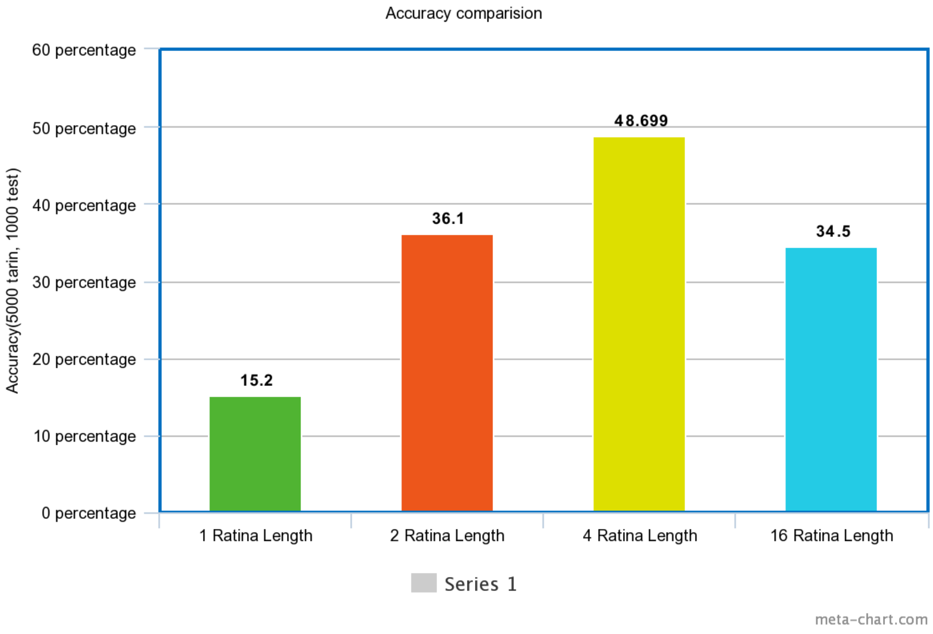




While making the code so that i can implement numba i have taken care of few thing like all the pixels of the pics must be covered in random pix selection to train a discriminator, the RAM address is int instead of str, I made a max function which takes the first max value of the discriminator as the right one. The accuracy increased significantly as now the accuracy while training the network with 5000 images and testing it by 1000 is 73.6 percent which in open source code was 55 percent.



If we vary the retina length the results(accuracies) are:



Now, the program is finished as I implement oops concept and make classes so that people can import it and make objects. The comparison of the timing of our numba implemented program with the open source is remaining.