

“cloudclassifyANNColor.py” is the contains the final version of the cloud classify class



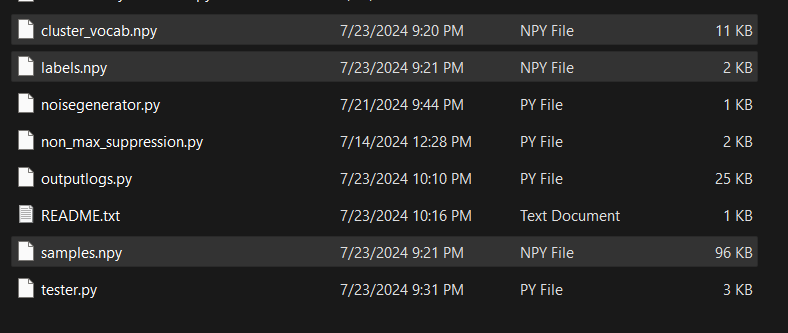
“tester.py” is the driver file used to run instances of the cloudclassifier class and output photos to file and output accuracy and other info to the console



This module contains a fast non maximal suppression algorithm provided by the authors of the opencv packt publishing book

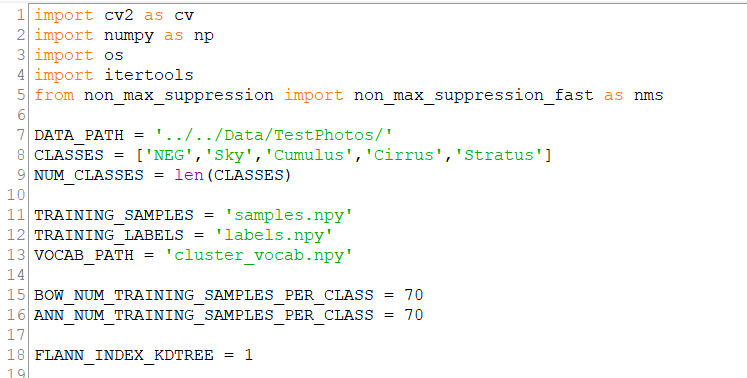


This is a script I used to generate instances of the “NEG” class of random noise to use as a negative detection. I wrote some other helper scripts the organize files and rename things that I either deleted or left in other folders.

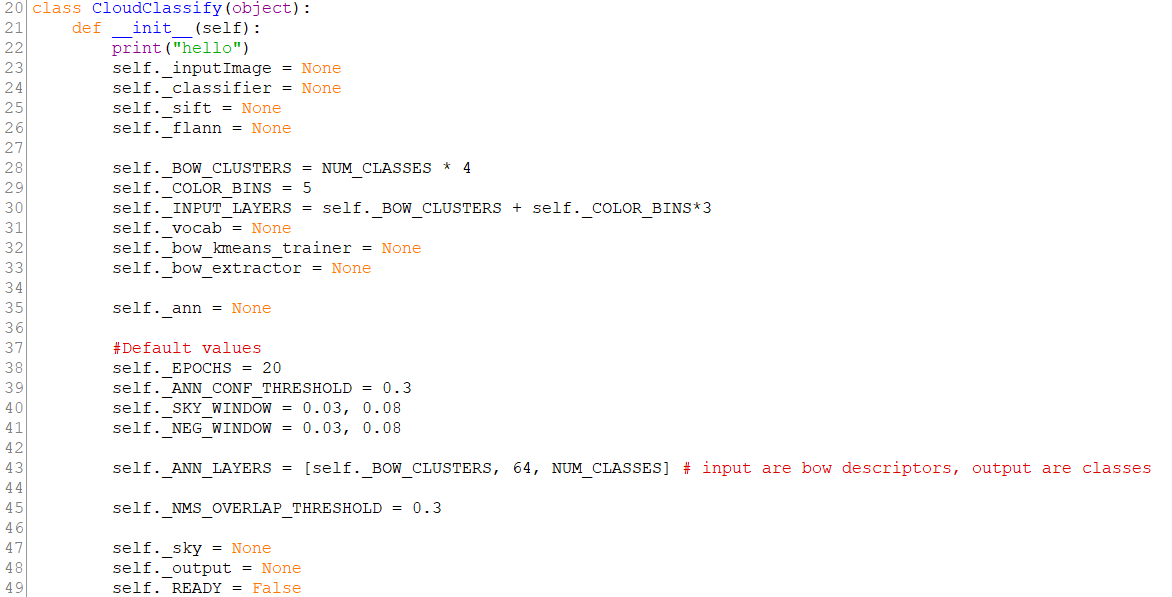


These three numpy files are generated by the code when it has access to the dataset. Deleting them will recluster and create a new vocab. I couldn’t get opencv’s built in ann to save an instance of itself to file as it is supposed to be capable of, however since it trains pretty quickly, I just had it save the samples of the BOW descriptors it trains off of to file, since actually extracting the descriptors was the limiting factor in prep time. Changing parameters in the test file usually requires deleting the numpy files, unless the BOW CLUSTERS weren’t changed.

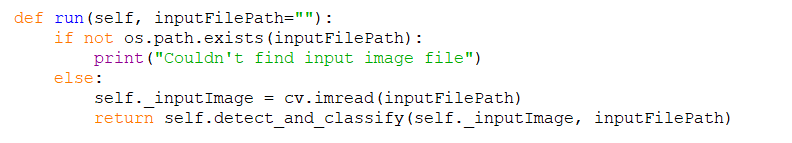
Lets look at cloudclassifyANNColor.py



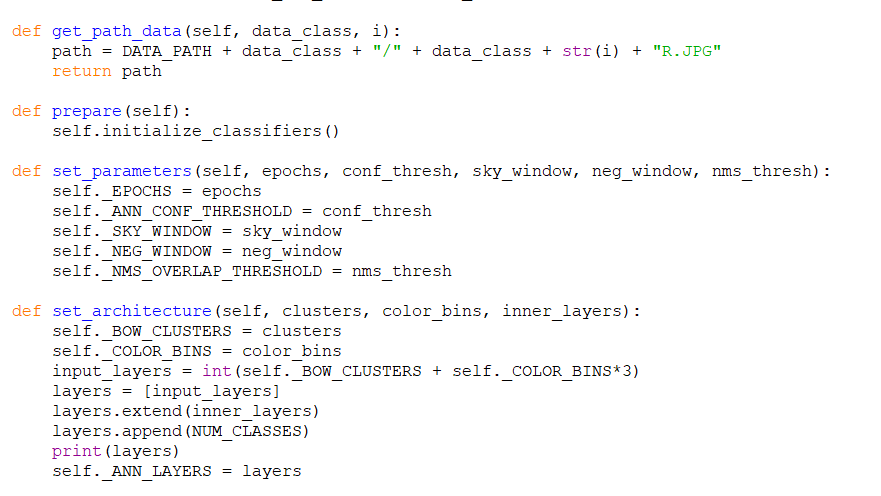
Here are all the macros and imports needed in the class.



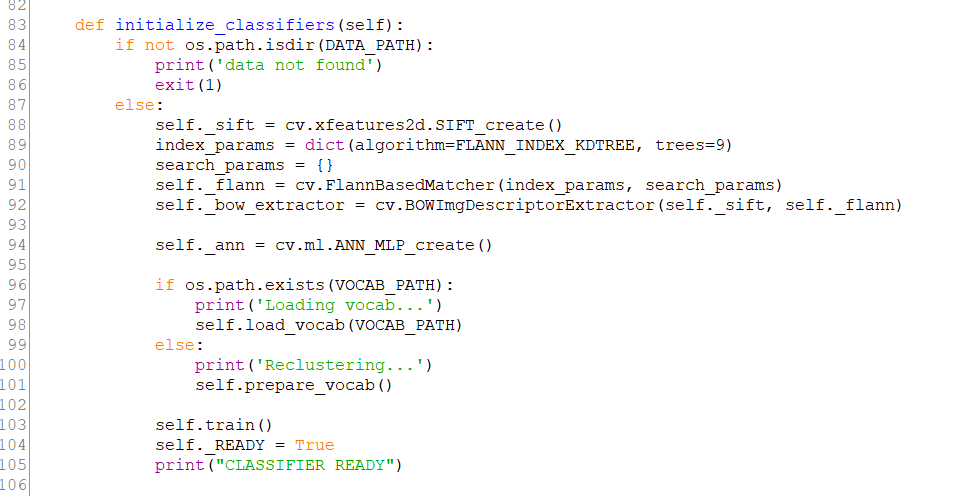
Here is an initializer and an initialization of all of the member variables with default values. These default values can be mutated by outside classes using the setters later in the file.



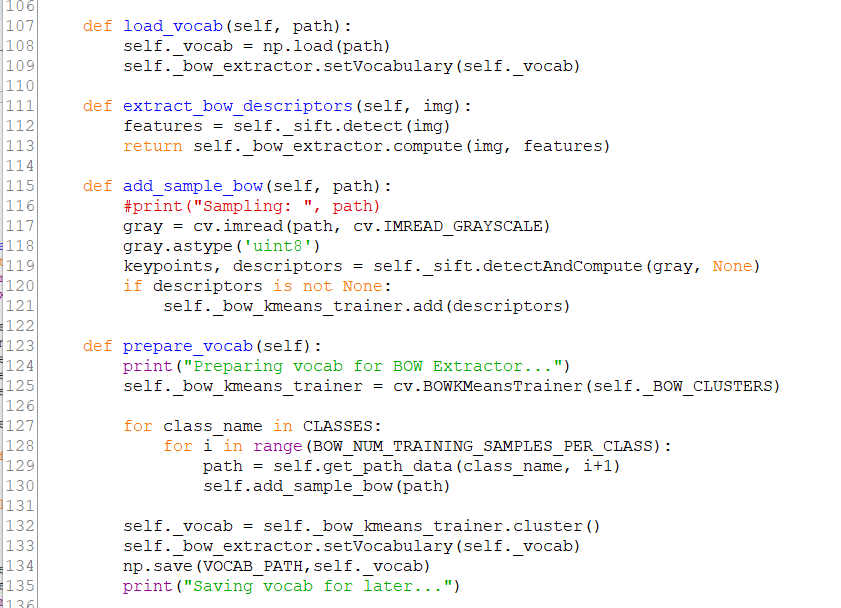
Here is the function for taking an input image by its file path and outputting a new image with the detection boxes drawn onto it along with the predominant cloud type. It is really a wrapper for detect\_and\_classify which does all the magic.



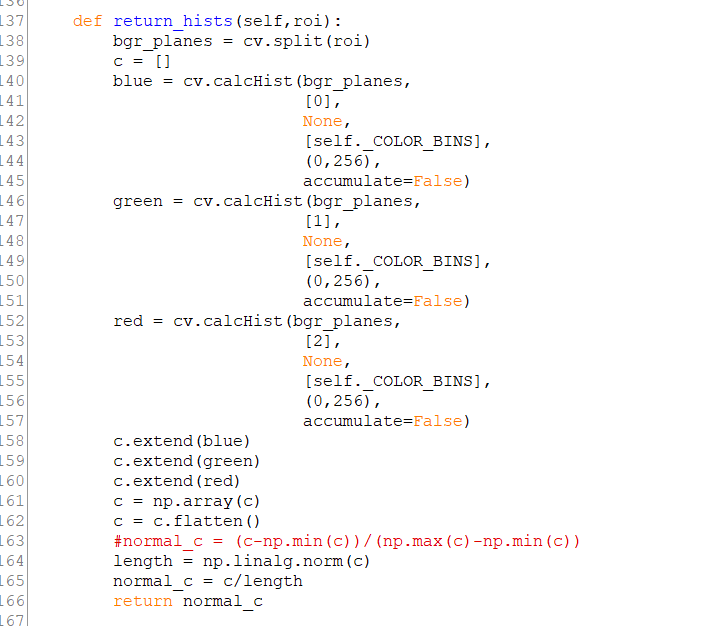
Here is a helper function and a wrapper for initializing all the different machine learning objects needed for the program like the BOW extractor and the ANN. There are also two setter functions for the driver file or other classes to use for easy parameter changing.



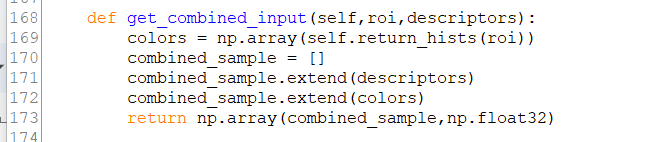
Here the feature detector and feature matcher are initialized for the BOW extractor along with the initialization of the ann. Then the vocab is prepared and the ANN is trained on it



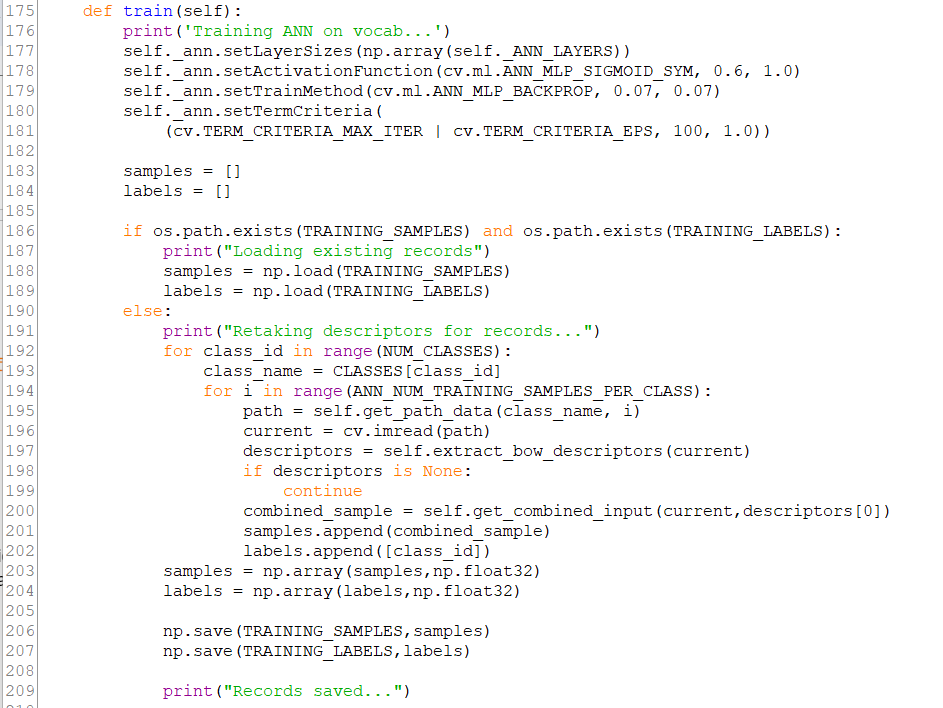
Here are some more helper functions for loading vocabulary, extracting BOW descriptors, adding a sample to the BOW extractor, and preparing the vocab if needed.



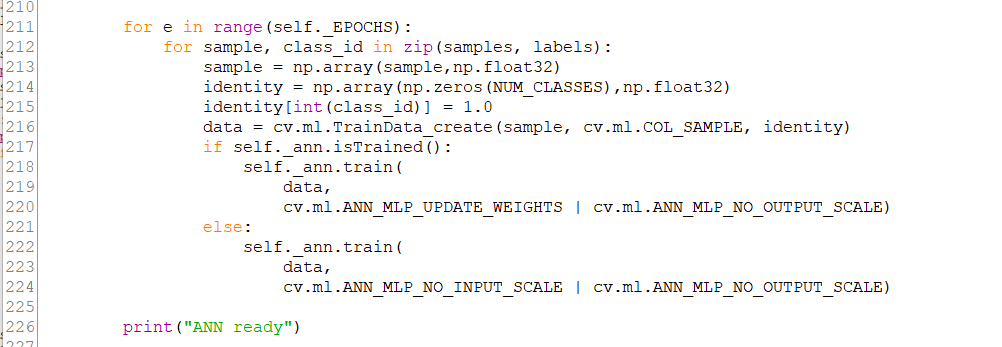
This helper function is used for taking a histogram along each channel of a ROI and turning it into a one dimensional normalized vector encoding all of that information for use in the ANN later.



This helper function takes in the BOW descriptors and the color vector and combines them into a one dimensional vector for use as input to the ANN.



This is part of the function used to train the ANN, all of the combined input vectors are taken from each file in the dataset. Training samples and labels are created and saved.

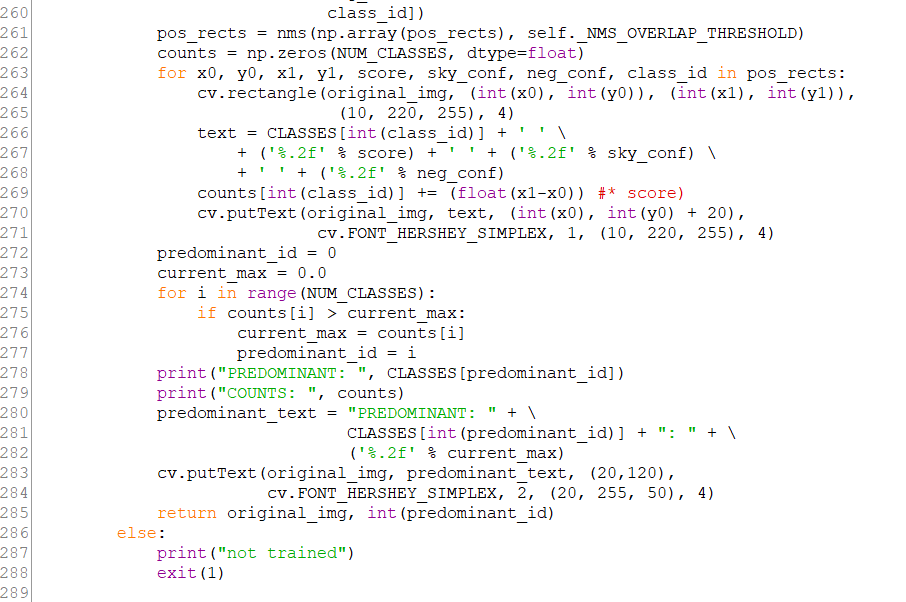


Then the ANN is trained based on the EPOCHS macro for a number of epochs.



This is the main functionality of the program. The input image is resized into a pyramid and for each image in the pyramid a sliding window goes along it, taking a combined input vector from the window then having the ANN get a prediction. IF the prediction meets a threshold and certain tolerances for the negative classes, then it is taken as a positive classification.

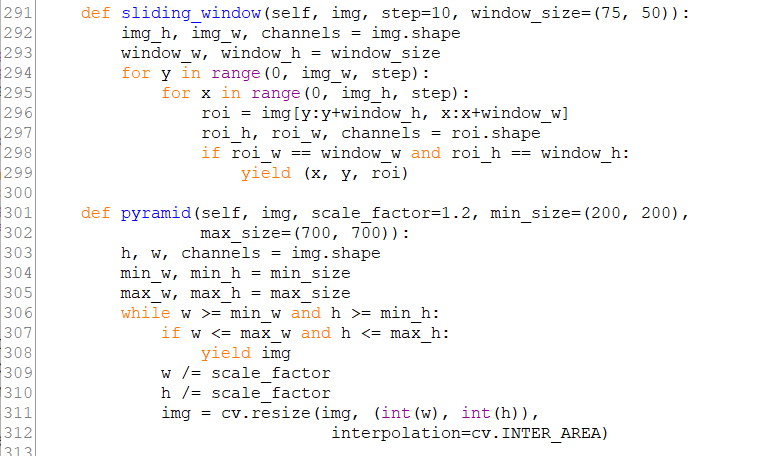
Then the positive rectangle’s coordinates and information are added to a list.



Then the positive rectangles are filtered by the NMS algorithm.

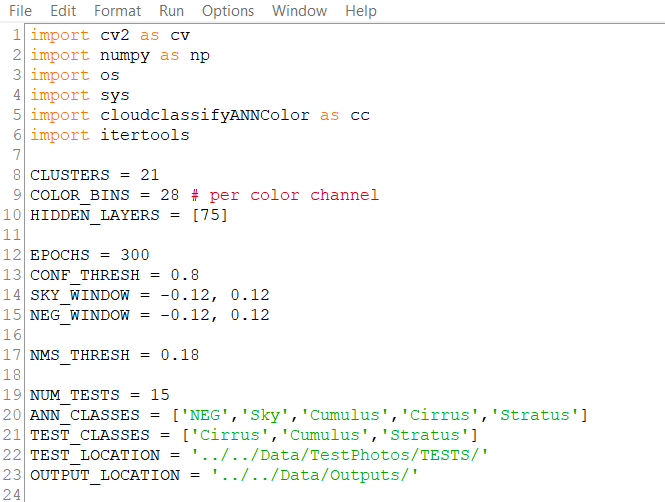
‘

Then every final positive rectangle is looped through and drawn onto the output image along with the confidence for the class, and confidences for the negative classes. A count of the area (using side length since it is a proportion of area and every sliding window has the same proportions) is used to keep an area score of each cloud type in the photo and the maximum is taken and returned as the predominant cloud type along with the output image.

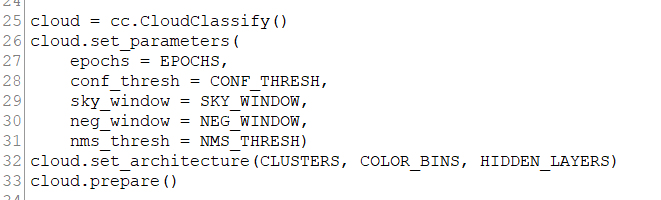


Lastly here are the helper functions for resizing the input and sliding a window across.

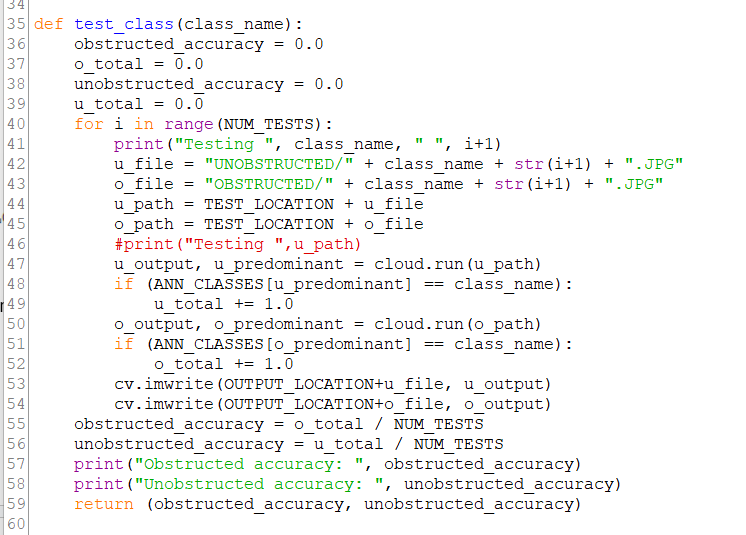
Now lets look at the “tester.py” file:



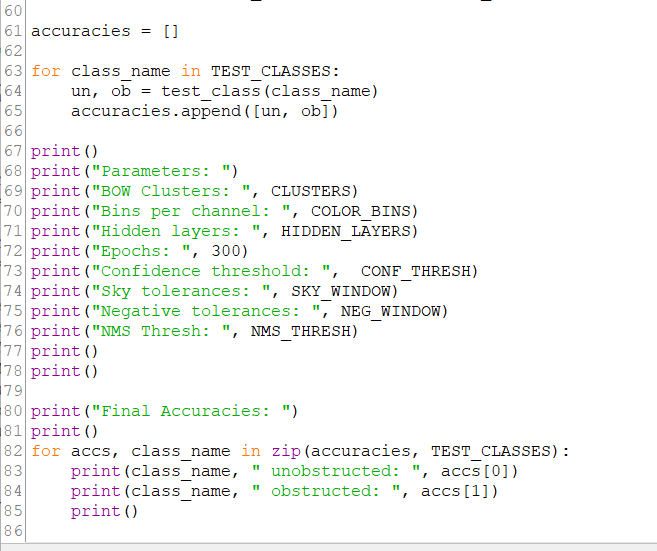
All of these macros are used to change parameters of the instance of cloudclassifyANNColor. Changing them yields difference results in accuracy for different classes and also speed.



Here the instance of the cloud classifier has its parameters and architecture set then it is prepared for running.



This is a helper function for testing all of the obstructed and unobstructed test photos for a given testing class.



Finally each class is tested and the results and information are output to the console.

Again, deleting the .npy files will allow you to change certain parameters and recluster and retake training samples. Once the necessary npy files are already there, the is much less overhead and the program goes right into getting results. If there is an problem in running the program, try deleting the npy files and let it retrain itself.