Author: Adam Witiak

Written: 15 April 2022

Edited: 3 December 2022

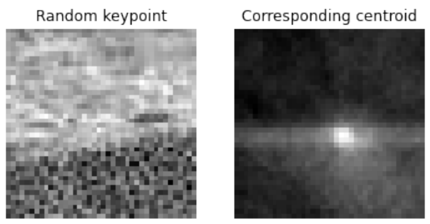
Image Classification Report

This project's task was to classify images into one of three categories using feature descriptors and a support vector machine (SVM).

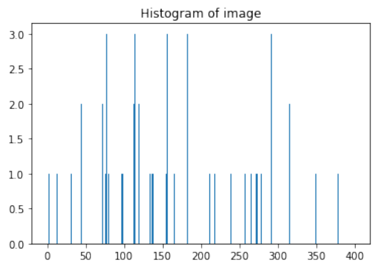
To start, I extracted feature descriptors from each image using OpenCV's SIFT algorithm. This gave me a list of descriptors. I had to select the same number of keypoints in each image as the image with the fewest keypoints . Initially, I truncated each input to make them all the same size. This approach was naïve, so then I tried selecting randomly from each image's keypoints. I later found out that OpenCV keypoints have a response value based on their strength, so I took the top keypoints based on that.

My approach to K-means was to create two lists: all descriptors and all keypoints. I would need the descriptors to get the clusters themselves and the keypoints to create histograms of the training data. I stacked all descriptors and keypoints, as well as assigning each keypoint a classifier that linked it to the image it was from. I used np.random to generate random indices for initial cluster centers. I also defined a method to check when cluster centers were no longer changing. I used broadcasting NumPy arrays to make calculating distances faster. Then I used np.argmin to find the closest cluster center. The descriptor and keypoint were put into lists, and if the clusters did not change, then these lists were returned. The algorithm is slow since it makes a lot of deep copies and is not JIT-ed. Improvements could be made with these changes.

Below is a random keypoint and the sum of all keypoints in its corresponding cluster (its centroid). There is a strong white dot in the middle of the centroid, likely where all the images in a cluster share a color. Note that SIFT descriptors are invariant to changes in scale and orientation. Many keypoints in the same cluster will look similar, but most of the colors will not line up due to these differences. This likely explains the drastic difference between the two images.

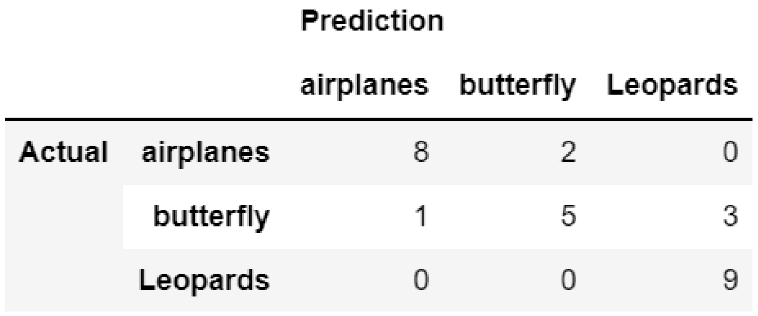


Within the K-means algorithm I created the histograms for the training data. I used the classifiers attached to the keypoints to find all keypoints from one image and assemble the histogram from it.

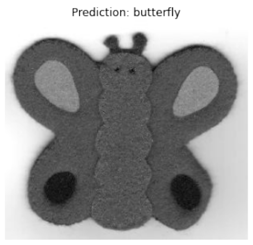


As you can see from the histogram, the bin with the most keypoints in it only has 3 keypoints. This is because each image was restricted to less than 70 keypoints, and there are 400 buckets.

I used Scikit-Learn's OneVsRestClassifier. This would create a one vs. all SVM. I passed in the histograms and corresponding output to train it. I used a very similar approach as earlier to find SIFT descriptors and calculate histograms of the test data. The only difference was I had to repurpose the code to calculate distance from the K-means algorithm to find what bin each descriptor in the test set belonged to. These histograms were used as the input for the prediction. Then I used the confusion_matrix method, also from Scikit-Learn, to generate the confusion matrix. As you can see, I ended up using butterfly images instead of dolphin images because there were more of them. In the following example, the butterfly and leopard were correctly identified, but the airplane was mistaken for a butterfly.



		Prediction		
		airplanes	butterfly	Leopards
Actual	airplanes	8	2	0
	butterfly	1	5	3
	Leopards	0	0	9



I found that this specific model had an accuracy of 78.571%. When I ran the algorithm other times, I was getting an accuracy percentage in the low 80's. From these iterations, it seems like the leopards had very different histograms compared to butterflies and airplanes; they were almost never misinterpreted as another classification. I suspect that with more images in the training set and analysis on choosing a good k for k-means (via gap statistics, for example) would yield better results.