CMSC 426 – Computer Vision

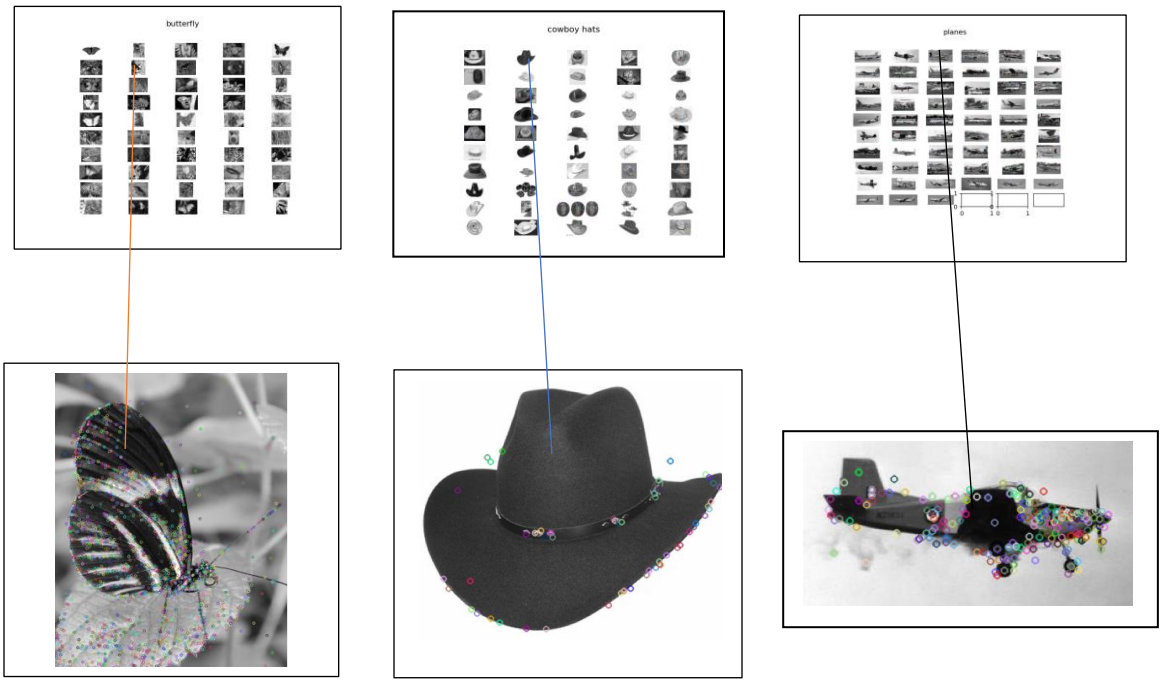
Project 2: Feature-based Image classification

This project aims to identify various butterfly, plane, and cowboy hat images to a high degree of accuracy through comparing data . The reader can find my thought processes, visual data, and overall results in the report below.

Alessandro Russo

3.1 – Find Sift Features:

To begin I utilized the sift methods ‘create()’ and ‘detectAndCompute()’ in order to find all key features of all training set images. The key points can be visualized by the multicolor circles around the images. I kept track of computers understanding of these features (known as descriptors) in lists then stacked vertically to better maintain them. I also kept track of each images individual count of features to divide into bins later.



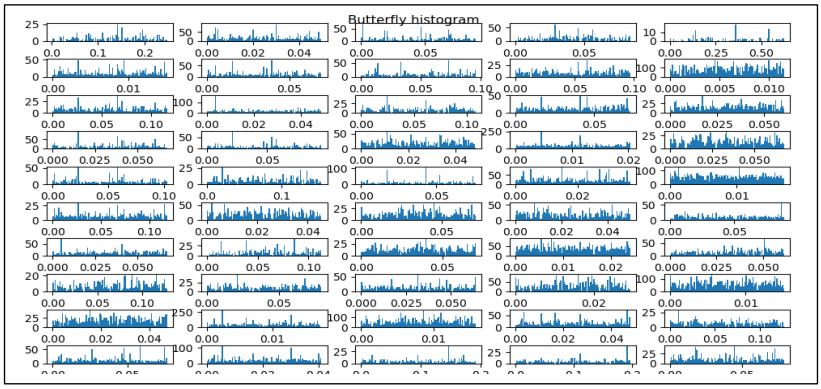
3.2 – Clustering:

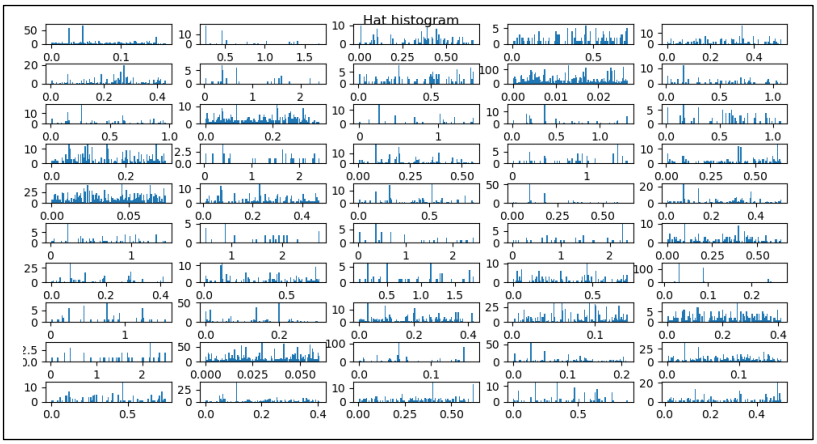
For this section I created three separate models that used the KMeans algorithm to predict the clusters of the three classes of descriptors and find the associated labels to these models. I made the number of clusters 100 and max iterations of 300 as a compromise for runtime and performance.

3.3 – Form Histograms

With the associated labels to each class, and three separate lists that denote the number of features per image I formed histograms with each bin representing a cluster. The normalization was the trickiest because I had to carefully divide each label by the number of features per image. These new labels were kept track of as well a stacked version of all training histograms. I found visual similarities in the classes which was a good sign.

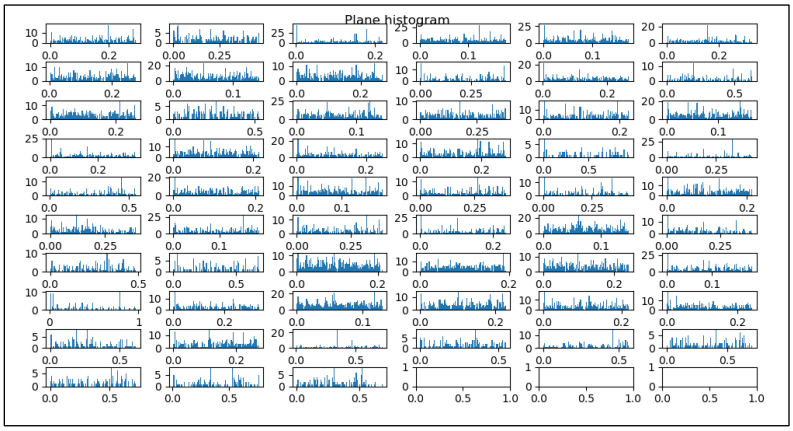
The butterfly training set histograms seem dense across the board for the most part.





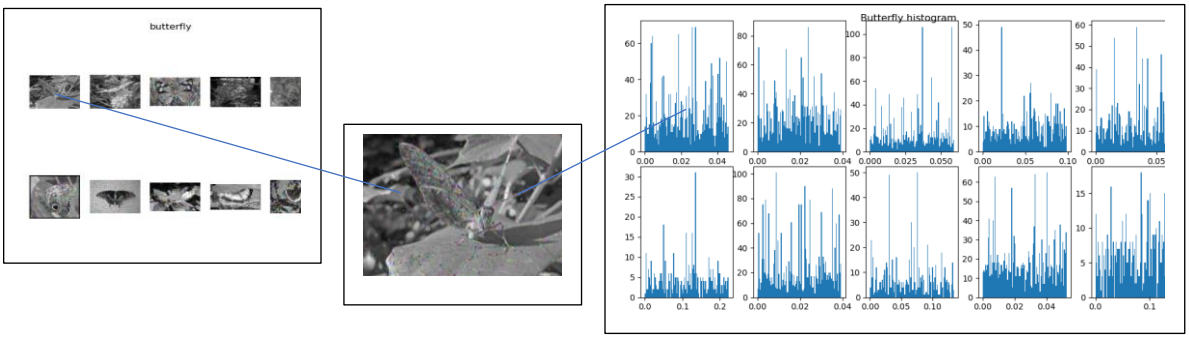
The cowboy hat training histograms were much more spacious across the board. They look very different to the collection of butterfly training images.

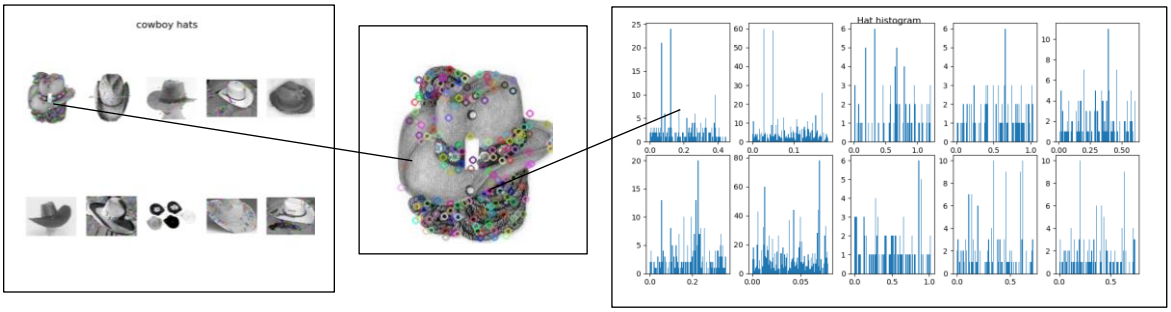
The plane histograms seem like a combination of both, however a lot of spikes occur around the same time. Overall the three different sets do have distinct visual features.

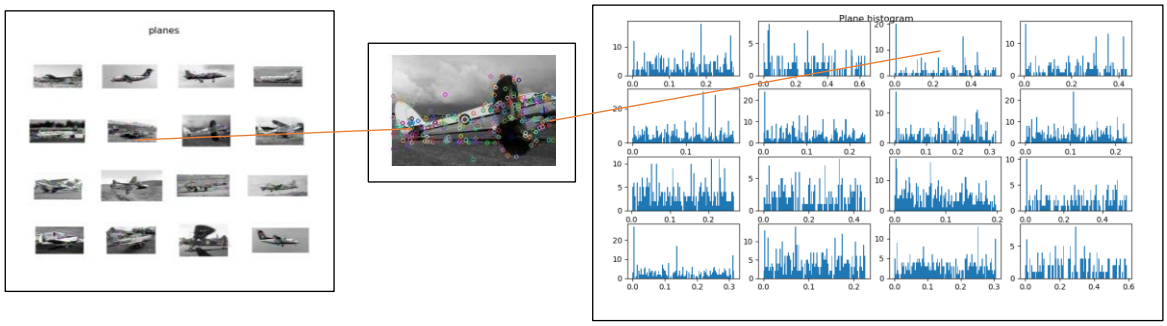


3.3 – Prepare for Classification

Now I dealt with the testing data where we find the sift features from each testing image and assign these features to clusters. Using the same model from clustering and the newly returned descriptors from ‘find_sift_features()’ I acquired new testing labels to generate the histograms.







3.5 – K = 1 Nearest Neighbor:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(all_img_train_histogram, all_img_train_classes)
predictions = knn.predict(all_img_test_histogram)
```

Using the KneighborsClassifier I was able to find the class an image belonged to by comparing the histogram vector to the closest one associated with test image. It was straight forward.

```
[[ 7  0  3]
 [ 1  4  5]
 [ 0  5 11]]
```

This is the confusion matrix presented at the end. The first row represents the butterfly testing images, the second is the cowboy hats test images and the third is the planes test images. The diagonal represents how many were guessed correct in that row. E.g. the butterflies were 80 percent correct, cowboy hats being 40 percent correct and the planes being 68.75%.

Using sklearns accuracy score metric the fraction correct was :

```
0.6111111111111112 or 61.1%
```

3.6 & 3.7– Linear Support Vector Machine

Similarly I did the same thing using sklearns Linear SVC and the non-linear SVC to try to improve these results. I was confused with my finding as the linear was somehow faster than the non-linear. The confusion matrices and results are below.

```
0.4722222222222222
[[ 1  0  9]
 [ 4  3  3]
 [ 3  0 13]]
0.6944444444444444
[[ 7  0  3]
 [ 1  8  1]
 [ 0  6 10]]
```

As you can see the Non-linear Linear was more accurate in guessing the correct class than the linear and my own implementation. This is interesting and I am relieved my code falls in between the two.