CS523 Computer Vision
Report

Image Classification with Bag of Features

Deniz Gokcin

May 31, 2020

Introduction

This report explains the implementation details of CS523 Computer Vision Assignment 3, which is about classifying a 4-class dataset with bag of features. The general idea behind a bag of visual words is to represent an image as a set of features. Features consist of keypoints and descriptors and no matter if an image is rotated, shrinked or expanded, the keypoints will always be the same. A descriptor is the description of a keypoint. The keypoints and descriptors are used to construct vocabularies and represent each image as a frequency histogram of features that are in the image. By using the frequency histogram, we can find and predict the category of an image.

Running the code

To run the code, you need to re-organize the dataset to look something similar to:

```
dataset
├── train
│   ├── airplanes
│   ├── cars
│   ├── faces
│   └── motorbikes
└── test
    ├── airplanes
    ├── cars
    ├── faces
    └── motorbikes
```

After modifying the directory structure, you should call main.py with the parameters that are needed to run the experiment you want. For example, the following snippet will set the feature extraction method to keypoints, will use kmeans, k=50 for the clustering algorithm.

```
python main.py --train_path dataset-modified/train --test_path
dataset-modified/test --no_clusters 50 --clustering_alg kmeans
--feature_extraction kp
```

Running the code

Feature Extraction and Description

Scale invariant feature transform(SIFT) is a feature detection algorithm, to detect and describe local features in images. In order to apply SIFT, I first extracted the features of the train images using two different methods. First, I detected keypoints in each image using sift.detectAndCompute. Than I constructed a keypoint array by iterating over each image image using two different step sizes, 15 and 10. After obtaining my keypoints, I used sift.compute to get the descriptors.

SIFT with keypoints, grid with step_size = 15, grid with step_size = 10

Dictionary Computation, Feature quantization and Histogram Calculation

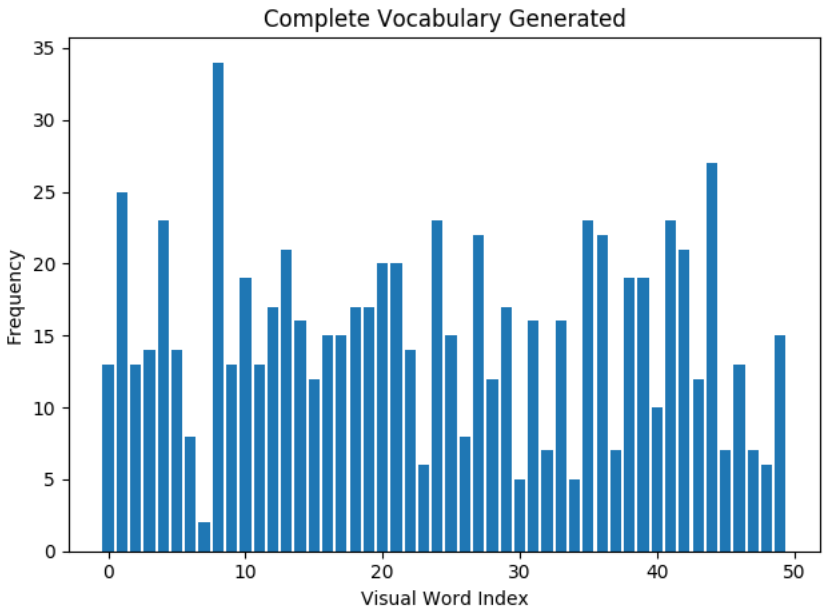
After I got my descriptors, I vertically stacked all of them into an array and send the vertically stacked descriptors to two different clustering algorithms. k-means and meanshift.

k-means clustering is a method of vector quantization, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

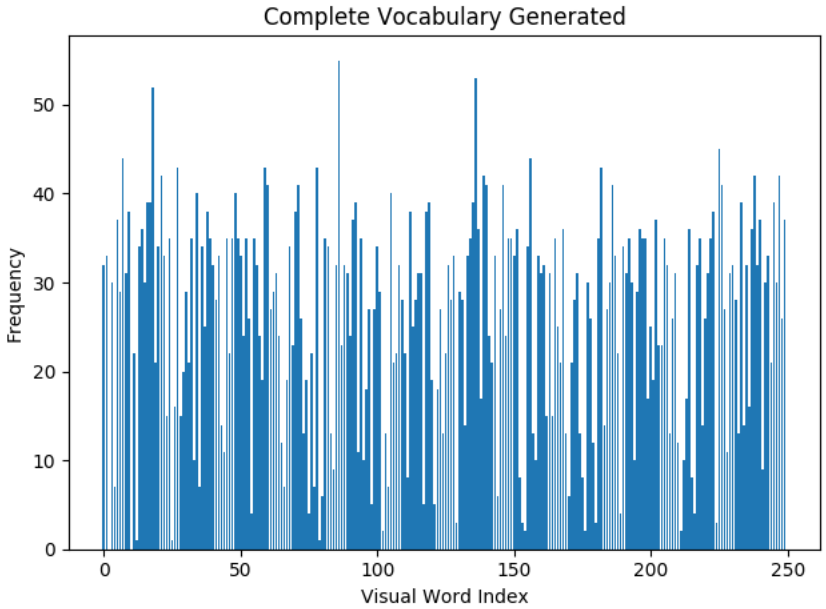
Mean shift algorithm is a non-parametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters.

Each cluster center produced by the clustering algorithms became a visual word. After obtaining the clusters, I created a histogram by calculating the number of occurrences for each visual word and ended up with the following vocabulary, for each experiment.

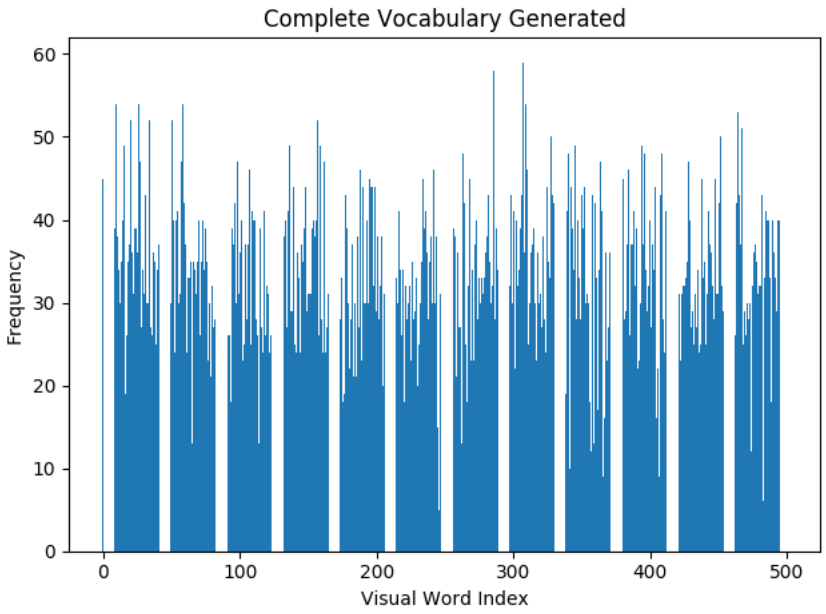
keypoints, k-means: k = 50



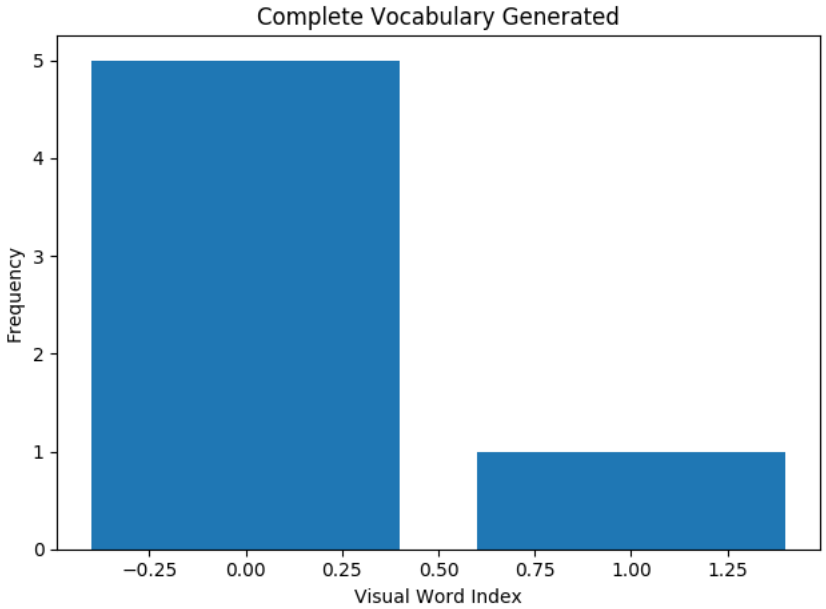
keypoints, k-means: k = 250



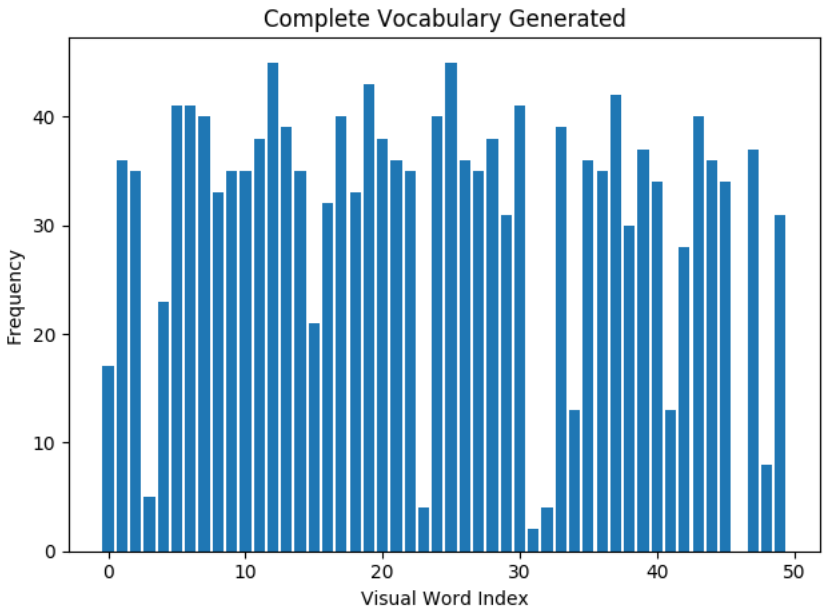
keypoints, k-means: k = 500



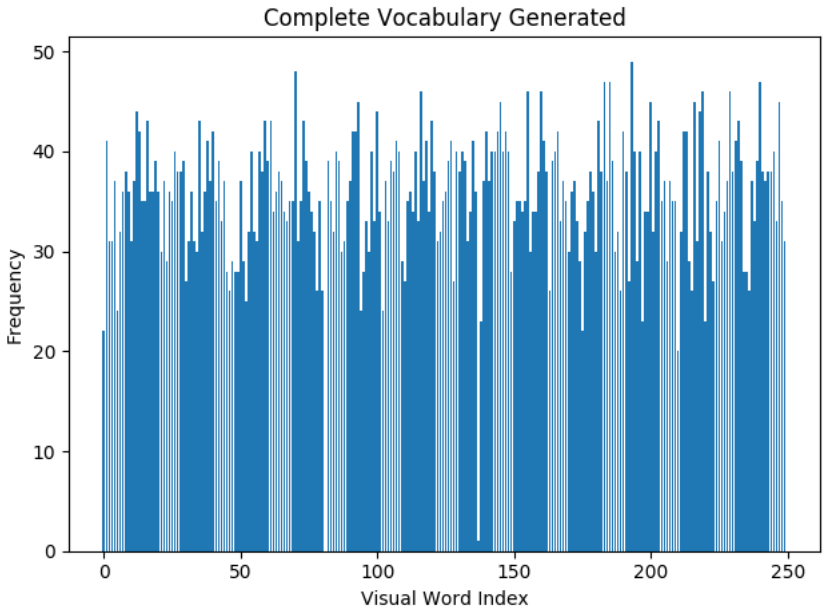
keypoints, k-means: Meanshift found k



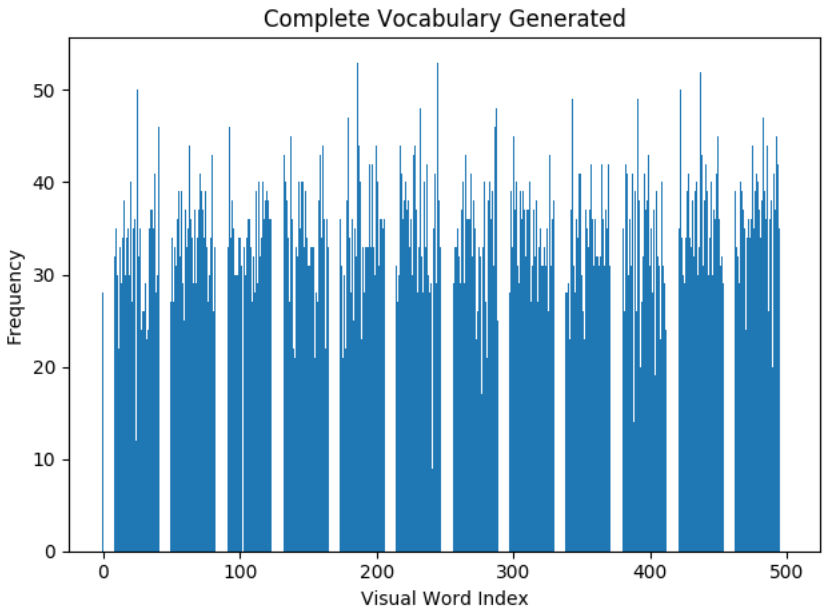
grid-1(step_size=15), k-means: k = 50



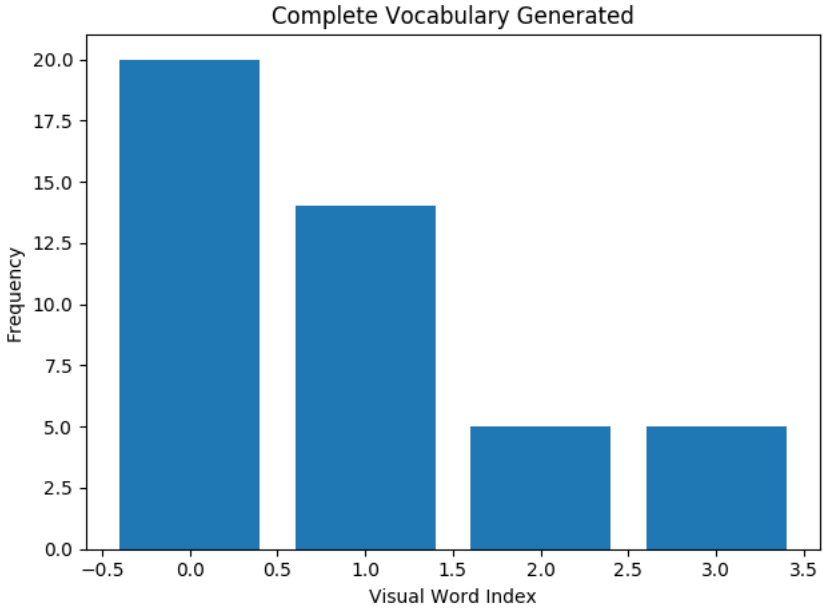
grid-1(step_size=15), k-means: k = 250



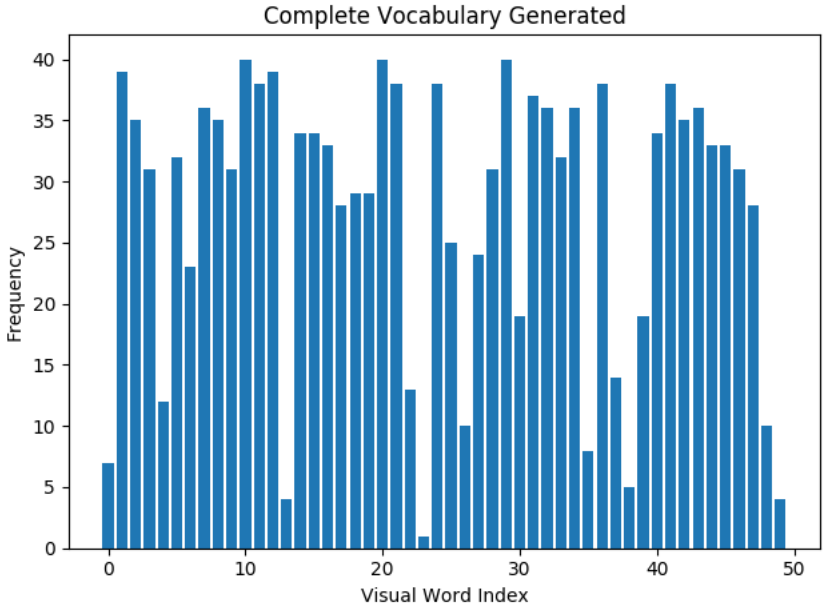
grid-1(step_size=15), k-means: k = 500



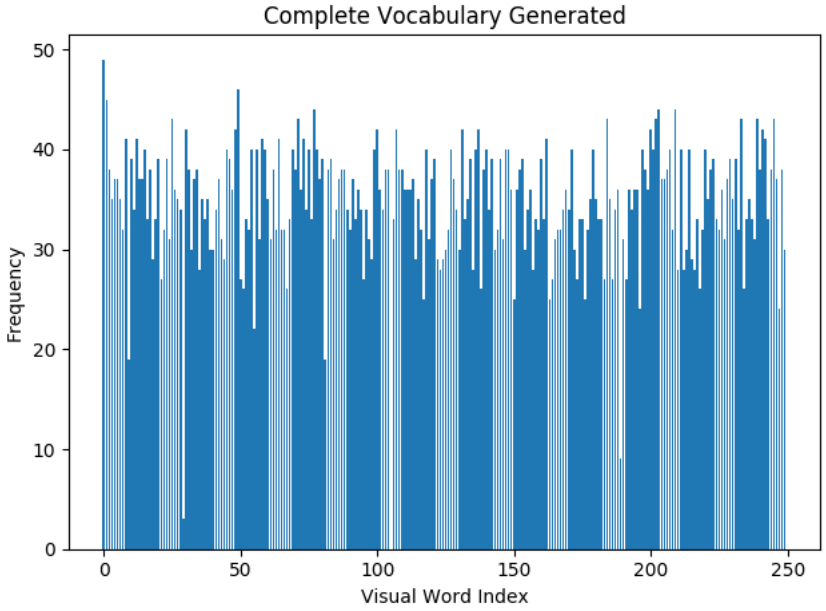
grid-1(step_size=15), k-means: Meanshift found k



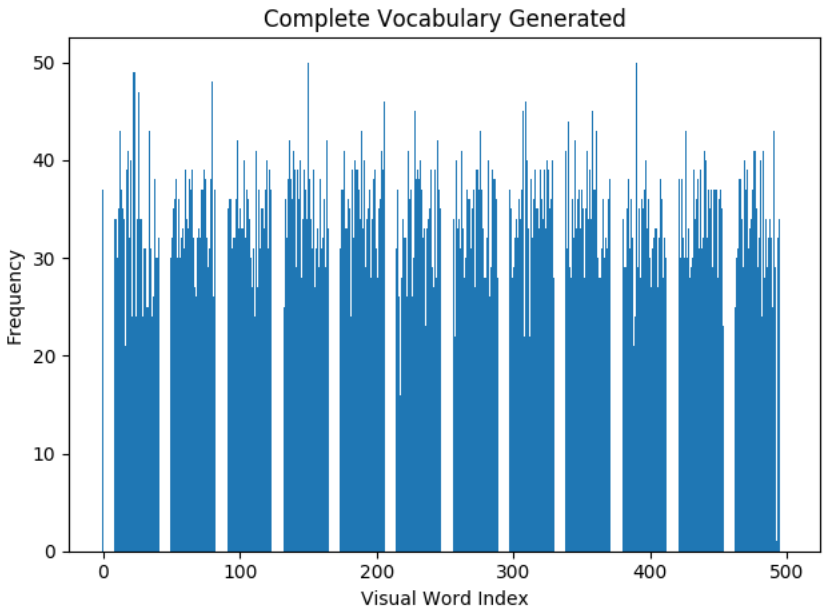
grid-2(step_size=10), k-means: k = 50



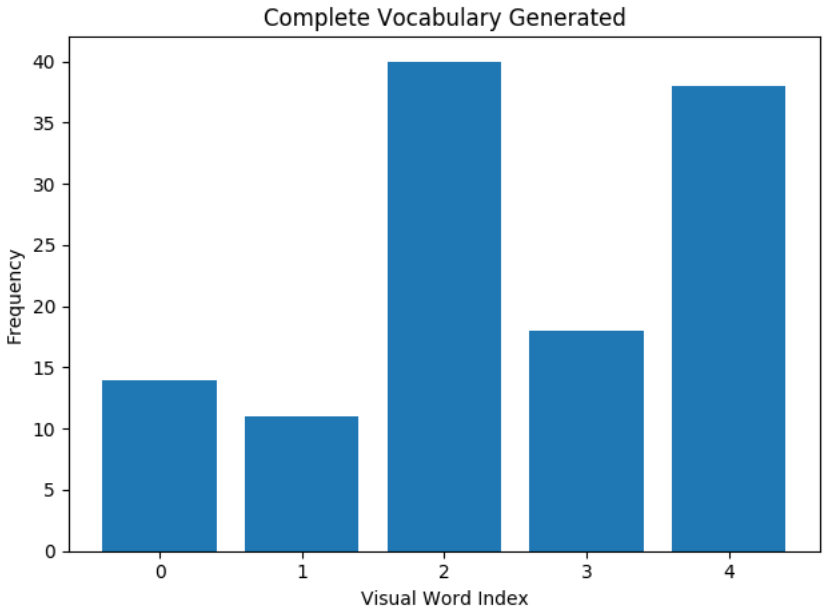
grid-2(step_size=10), k-means: k = 250



grid-2(step_size=10), k-means: k = 500



grid-2(step_size=10), k-means: Meanshift found k

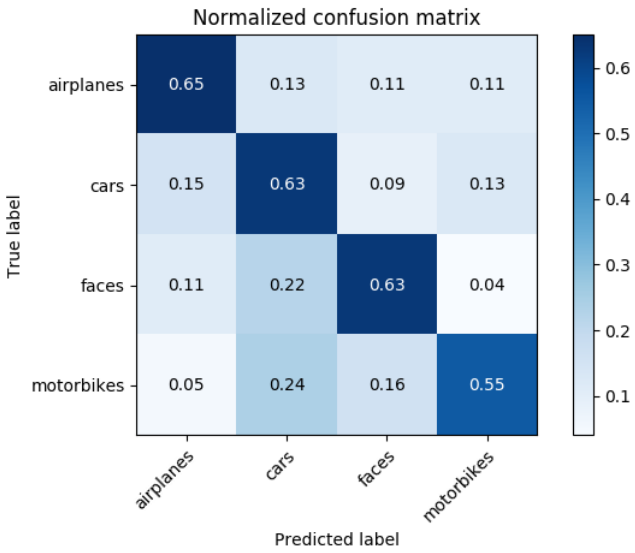


Classifier Training and Results

After obtaining a histogram, I normalized the data that I have using a StandardScaler. I fitted the normalized histogram to a support vector machine and completed the training.

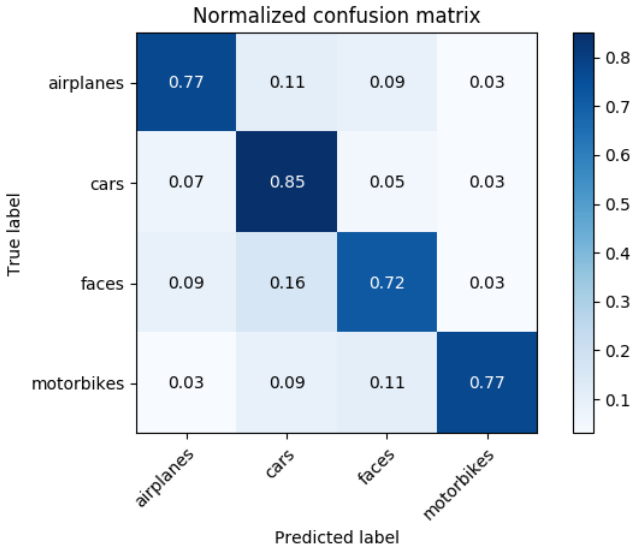
Below, there are the predictions for each experiment.

keypoints, k-means: k = 50



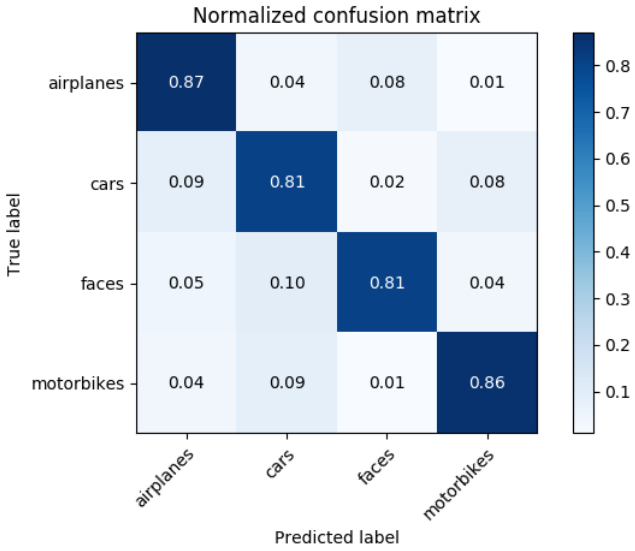
Accuracy: %66

keypoints, k-means: k = 250



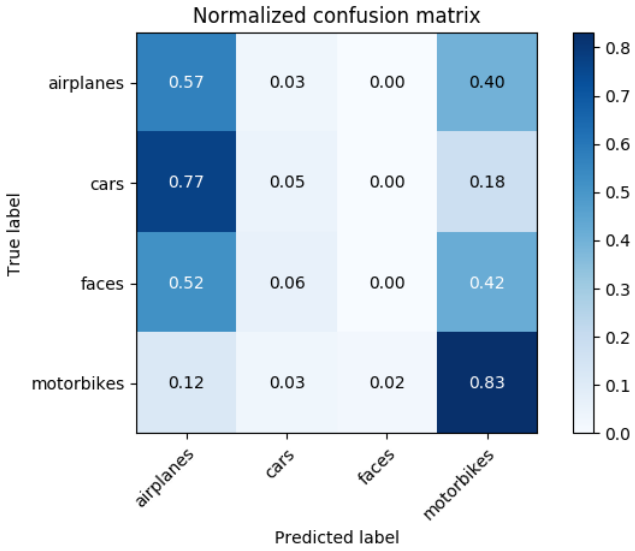
Accuracy: %77.7

keypoints, k-means: k = 500



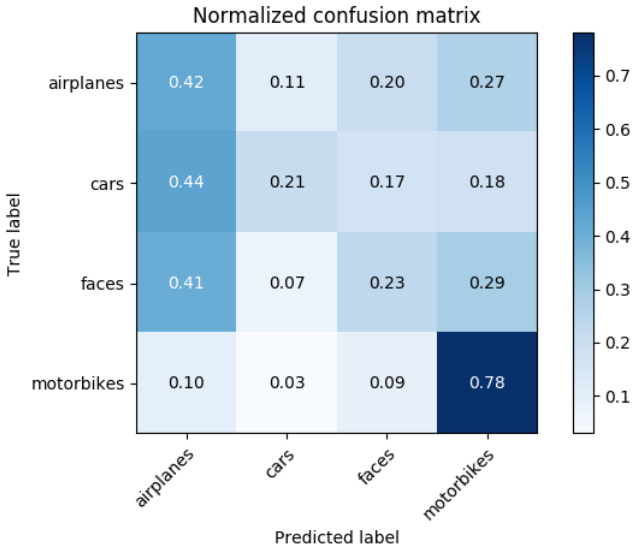
Accuracy: %83.8

keypoints, k found by Mean Shift



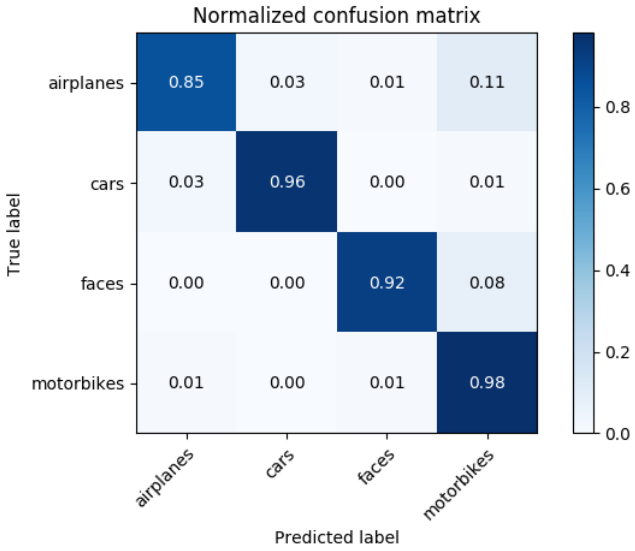
Accuracy: %36.2
Clusters Found: 2

keypoints, k-means: Meanshift found k



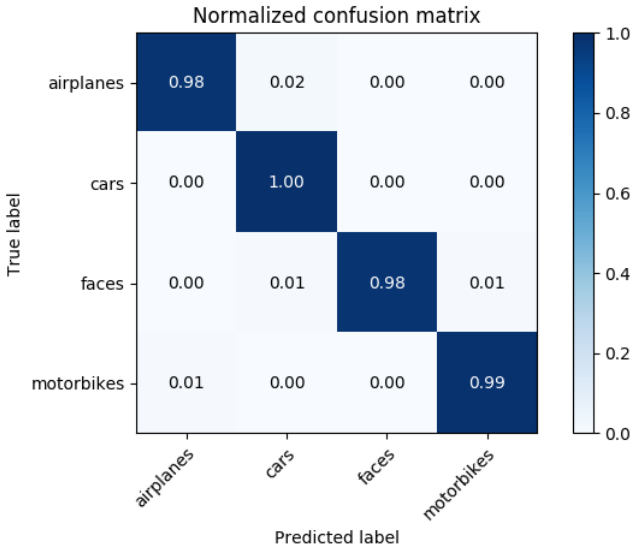
Accuracy: %41
Meanshift found k: 2

grid-1(step_size=15), k-means: k = 50



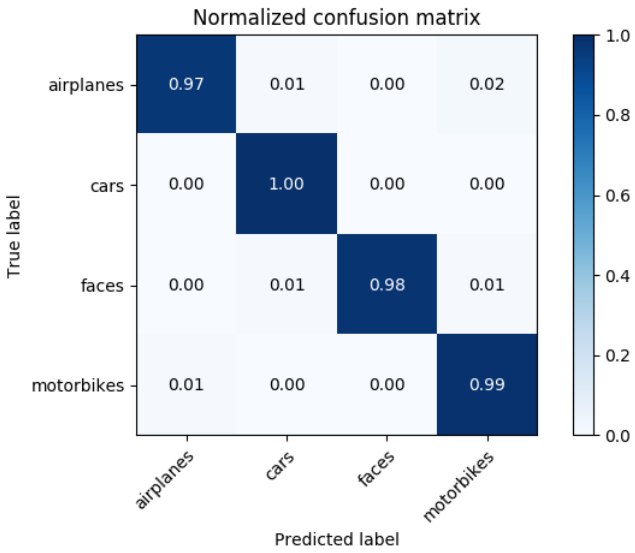
Accuracy: %92.7

grid-1(step_size=15), k-means: k = 250



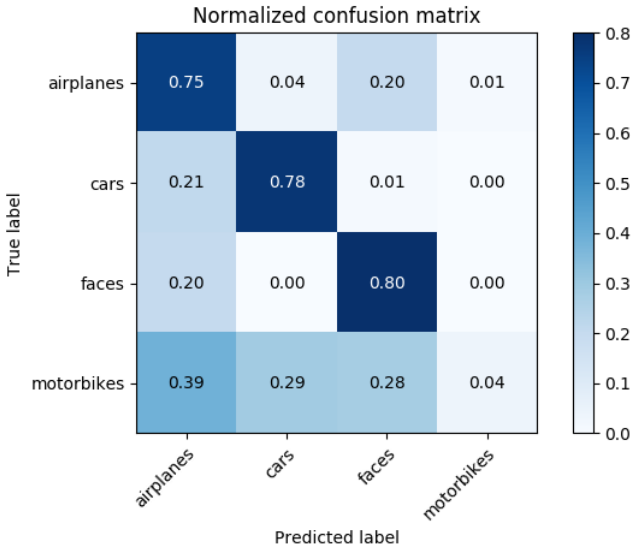
Accuracy: %99

grid-1(step_size=15), k-means: k = 500



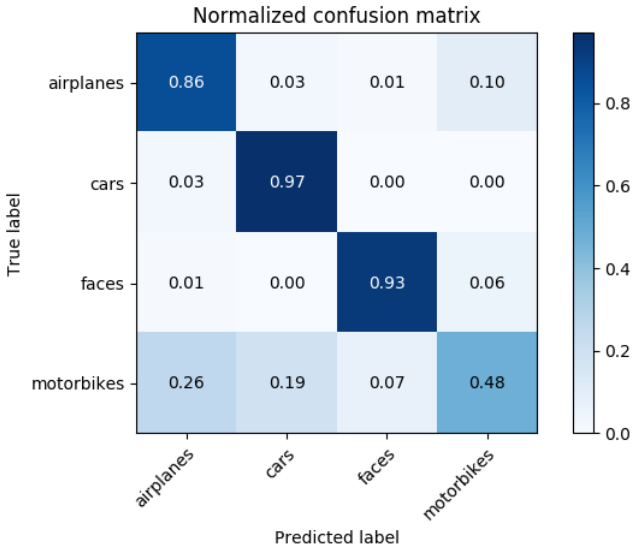
Accuracy: %98.5

grid-1(step_size=15), k found by Mean Shift



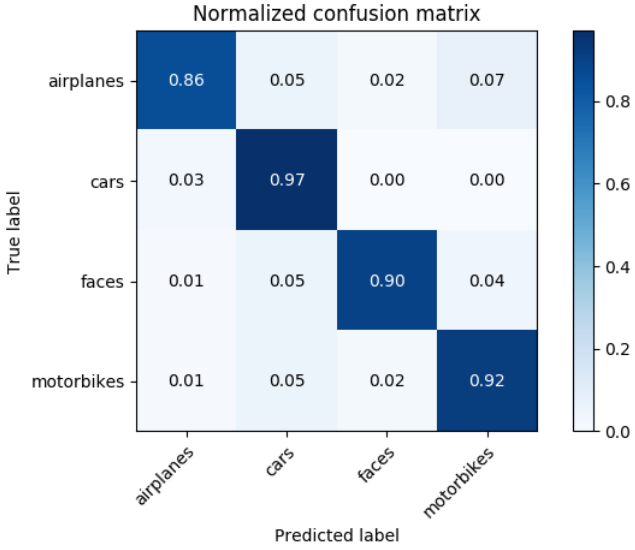
Accuracy: %65
Clusters Found: 4

grid-1(step_size=15), k-means: Meanshift found k



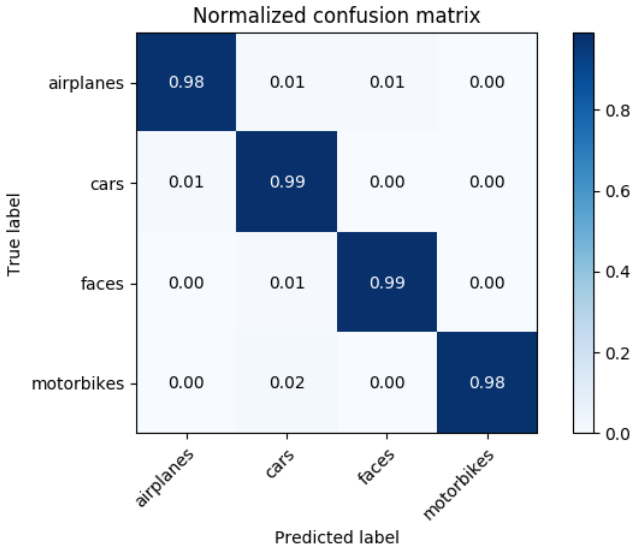
Accuracy: %81
Meanshift found k: 4

grid-2(step_size=10), k-means: k = 50



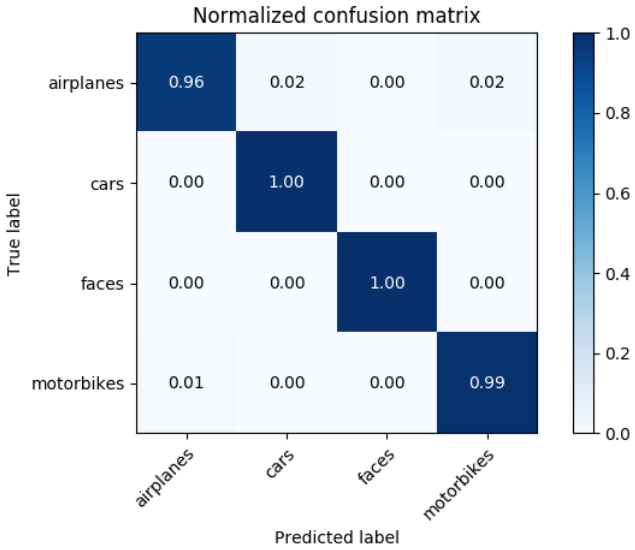
Accuracy: %91.2

grid-2(step_size=10), k-means: k = 250



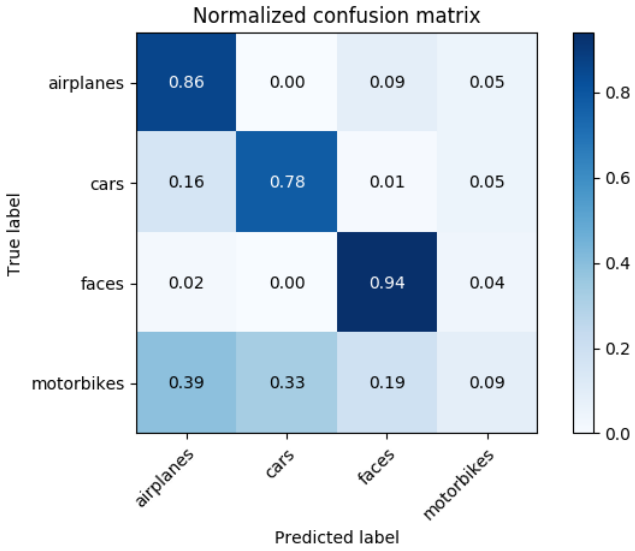
Accuracy: %98.5

grid-2(step_size=10), k-means: k = 500



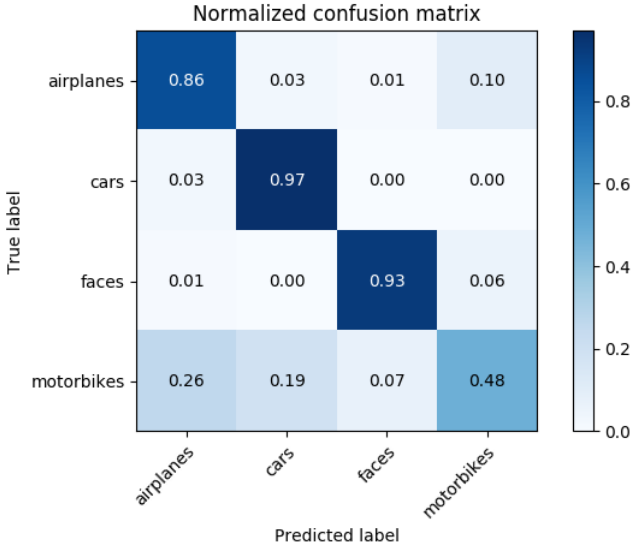
Accuracy: %98.8

grid-2(step_size=10), k found by Mean Shift



Accuracy: %66.7
Clusters Found: 5

grid-2(step_size=10), k-means: Meanshift found k



Accuracy: %74.5
Meanshift found k: 5

Comments

I can say that the k-means clustering algorithm did a great job when the step size for the grid was equal to 15 and k was equal to 250. I ended up with a 99% accuracy, which I think is quite impressive.

I was expecting more from the mean shift algorithm since it is a powerful algorithm and it took hours to complete the analysis. But for some reason, even if it took hours, the highest amount of clusters I found with meanshift was 5.

With this final assignment, I had a chance to use the cloud to do compute intensive tasks such as running meanshift with the default parameters.