Image Classification with Bag of Features

CS-423 Assignment 3 Report

Spring 2020

1. Introduction

1. Description

i. This assignment is about using bag of features image classification technique to represent images as a set of features. The first step is to use sift to extract features from a given image through grids or simply using key points. Considering our database which consist of frequent changes to orientation and illumination sift is an obvious choice because it is invariant to these features. The second step is to prepare vocabulary using sift descriptors to cluster sift features using k-means or mean shift clustering methods. The next step is to quantize vocabulary data using nearest neighbor. The last step is to train SVM using image features.

2. Steps

i. Feature Extraction (Grid, Key points)
ii. Feature Description (SIFT)
iii. Vocabulary Calculation (K means, mean shift => Nearest neighbor)
iv. Classifying (SVM)

2. Algorithms

1. SIFT (Scale Invariant Feature Transform)[1]

i. Description

SIFT transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection. The scale-invariant features are efficiently identified by using a staged filtering approach. The first stage identifies key locations in scale space by looking for locations that are maxima or minima of a difference-of-Gaussian function. Each point is used to generate a feature vector that describes the local image region sampled relative to its scale-space coordinate frame. The SIFT keys derived from an image are used in a nearest-neighbor approach to indexing to identify candidate object models. Collections of keys that agree on a potential model pose are first identified through a Hough transform hash table, and then through a least-squares fit to a final estimate of model parameters. When at least 3 keys agree on the model parameters with low residual, there is strong evidence for the presence of the object. Since there may be dozens of SIFT keys in the

image of a typical object, it is possible to have substantial levels of occlusion in the image
and yet retain high levels of reliability.

ii. Key localization

To identify locations in image scale space that are
invariant with respect to image translation, scaling, and rotation, and are minimally affected by noise and small distortions. To achieve rotation invariance and a high level of efficiency, select key locations at maxima and minima of a difference of Gaussian function applied in scale space. This can be computed by building an image pyramid with resampling between each level. It locates key points at regions and scales of high variation, making these locations particularly stable for characterizing the image. As the 2D Gaussian function is separable, its convolution with the input image can be efficiently computed by applying two passes of the 1D Gaussian function in the horizontal and vertical directions:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

For key localization, all smoothing operations are done using $\sigma = \sqrt{2}$
The input image is first convolved with the Gaussian function using $\sigma = \sqrt{2}$ to give an image A. This is then repeated a second time with a further incremental smoothing of $\sigma = \sqrt{2}$ to give a new image, B, which now has an effective smoothing of $\sigma = 2$. The difference of Gaussian function is obtained by subtracting image B from A, resulting in a ratio of $\frac{2}{\sqrt{2}} = \sqrt{2}$ between the two Gaussians. To generate the next pyramid level, resample the already smoothed image B using bilinear interpolation with a pixel spacing of 1.5 in each direction. The 1.5 spacing means that each new sample will be a constant linear combination of 4 adjacent pixels. Maxima and minima of this scale-space function are determined by comparing each pixel in the pyramid to its neighbors. First, a pixel is compared to its 8 neighbors at the same level of the pyramid. If it is a maxima or minima at this level, then the closest pixel location is calculated at the next lowest level of the pyramid, taking account of the 1.5 times resampling. If the pixel remains higher (or lower) than this closest pixel and its 8 neighbors, then the test is repeated for the level above. Since most pixels will be eliminated within a few comparisons, the cost of this detection is small and much lower than that of building the pyramid.

iii. SIFT key stability

To characterize the image at each key location, the smoothed image A at each level of the pyramid is processed to extract image gradients and orientations. At each pixel, A_{ij}, the image gradient magnitude, M_{ij} , and orientation, R_{ij} , are computed using pixel differences:

$$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i,j+1})^2}$$

$$R_{ij} = \operatorname{atan2}(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij})$$

The pixel differences are efficient to compute and provide
enough accuracy due to the substantial level of previous
smoothing. The effective half-pixel shift in position is compensated for when determining key location
For each key point, pixel sampling from the pyramid level at which the key was detected. The pixels that fall in a circle of radius 8 pixels around the key location are inserted into the orientation planes. The orientation is measured relative to that of the key by subtracting the key's orientation. The blurring is achieved by allocating the gradient of each pixel among its 8 closest neighbors in the sample grid, using linear interpolation in orientation and the two spatial dimensions.

iv. Indexing and Matching

Store the SIFT keys for sample images and then identify matching keys from new images. The k-d tree algorithm called the best-bin-first search method can identify the nearest neighbors with high probability using only a limited amount of computation. To further improve the efficiency of the best-bin-first algorithm, the SIFT key samples generated at the larger scale are given twice the weight of those at the smaller scale. This means that the larger scale is in effect able to filter the most likely neighbors for checking at the smaller scale. use the Hough transform to search for keys that agree upon a model pose. Each model key in the database contains a record of the key's parameters relative to the model coordinate system. Create an entry in a hash table predicting the model location, orientation, and scale from the match hypothesis.

v. Solutions for Affine parameters

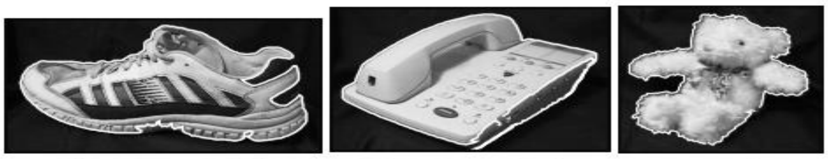
The hash table is searched to identify all clusters of at least 3 entries in a bin, and the bins are sorted into decreasing order of size. Each such cluster is then subject to a verification procedure in which a least-squares solution is performed for the affine projection parameters relating the model to the image. The affine transformation of a model point can be written as

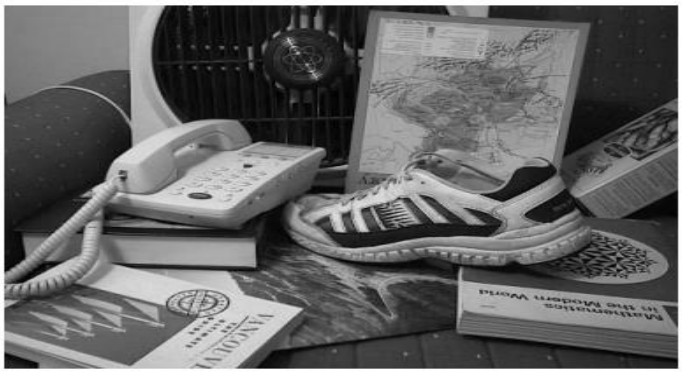
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

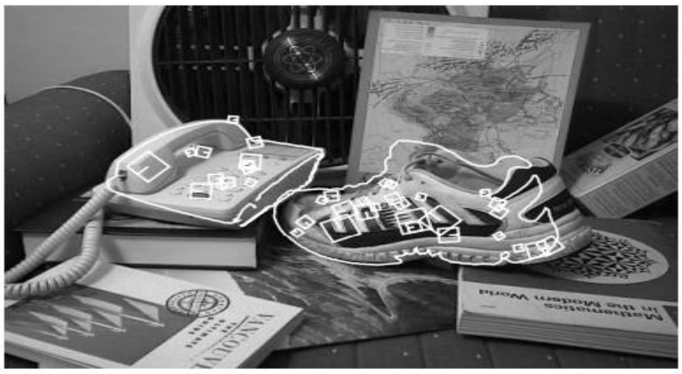
solve for the transformation parameters, so the equation above can be rewritten as

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

This equation shows a single match, but any number of further matches can be added, with each match contributing two more rows to the first and last matrix. At least 3 matches are needed to provide a solution. An example is as follows







vi. Sift Stages summarized[2]

Stage	Description
1.	Compute the Gaussian scale-space in: **u** image out:**v** scale-space
2.	Compute the Difference of Gaussians (DoG) in: **v** scale-space out: **w** DoG
3.	Find candidate keypoints (3D discrete extrema of DoG) in: **w** DoG out: $\{(x_d, y_d, \sigma_d)\}$ list of discrete extrema (position and scale)
4.	Refine candidate keypoints location with sub-pixel precision in: **w** DoG and $\{(x_d, y_d, \sigma_d)\}$ list of discrete extrema out: $\{(x, y, \sigma)\}$ list of interpolated extrema
5.	Filter unstable keypoints due to noise in: **w** DoG and $\{(x, y, \sigma)\}$ out: $\{(x, y, \sigma)\}$ list of filtered keypoints
6.	Filter unstable keypoints laying on edges in: **w** DoG and $\{(x, y, \sigma)\}$ out: $\{(x, y, \sigma)\}$ list of filtered keypoints
7.	Assign a reference orientation to each keypoint in: $(\partial_m \mathbf{v}, \partial_n \mathbf{v})$ scale-space gradient and $\{(x, y, \sigma)\}$ list of keypoints out: $\{(x, y, \sigma, \theta)\}$ list of oriented keypoints
8.	Build the keypoints descriptor in: $(\partial_m \mathbf{v}, \partial_n \mathbf{v})$ scale-space gradient and $\{(x, y, \sigma, \theta)\}$ list of keypoints out: $\{(x, y, \sigma, \theta, \mathbf{f})\}$ list of described keypoints

2. K means

i. Description[3]

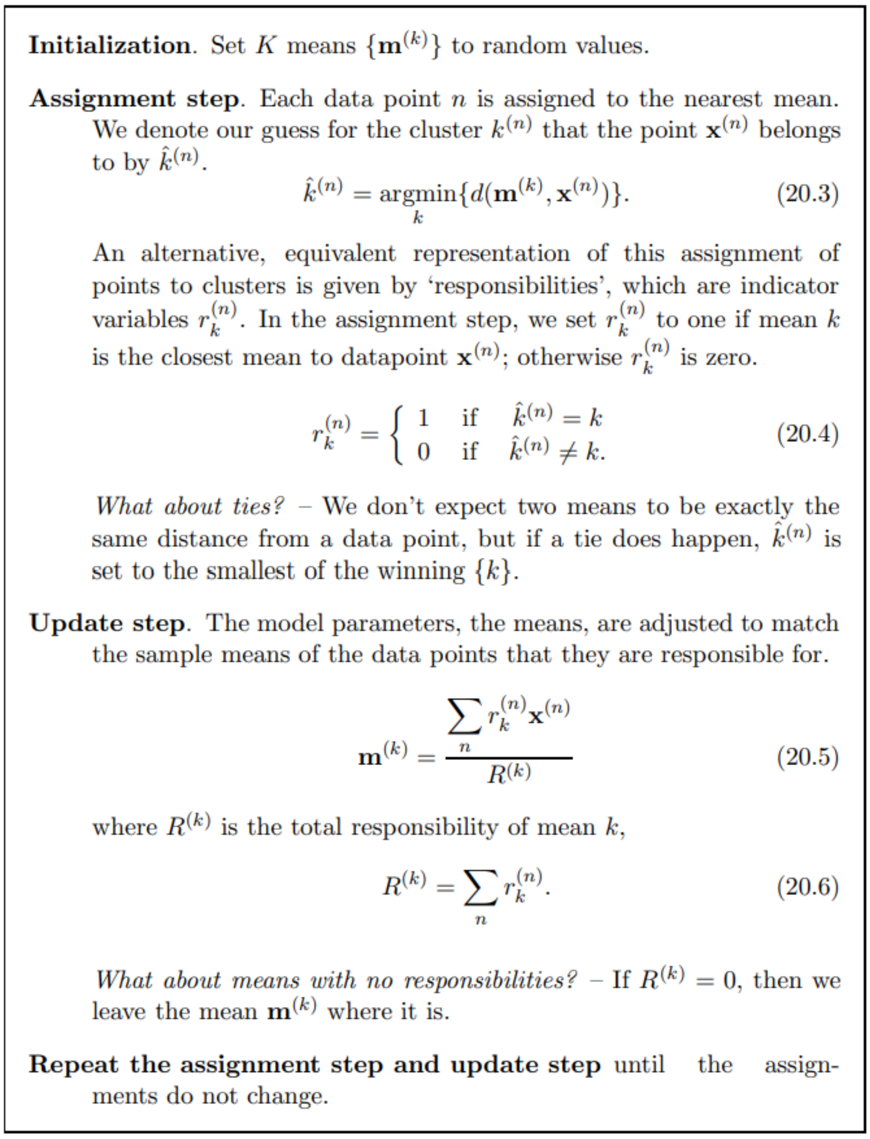
partitioning an N-dimensional population into k sets on the basis of a sample is called k-means. the k-means procedure is easily programmed and is computationally economical, so that it is feasible to process very large samples on a digital computer. In general, the k-means procedure will not converge to an optimal partition.

ii. K means Clustering[4]

The K-means algorithm is an algorithm for putting N data points in an I-dimensional space into K clusters. Each cluster is parameterized by a vector $m^{(k)}$ called its mean the data points will be denoted by $\{x^n\}$ where the superscript *n* runs from 1 to the number of data points *N*. Each vector **x** has *I* components x_i. We will assume that the space that x lives in is a real space and that we have a metric that defines distances between points, for example

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_i (x_i - y_i)^2.$$

start the K-means algorithm to random values. K-means is then an iterative two-step algorithm. In the assignment step, each data point n is assigned to the nearest mean. In the update step, the means are adjusted to match the sample means of the data points that they are responsible for.
A pseudo code is as follows



Initialization. Set K means $\{\mathbf{m}^{(k)}\}$ to random values.

Assignment step. Each data point n is assigned to the nearest mean. We denote our guess for the cluster $k^{(n)}$ that the point $\mathbf{x}^{(n)}$ belongs to by $\hat{k}^{(n)}$.

$$\hat{k}^{(n)} = \underset{k}{\operatorname{argmin}}\{d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\}. \tag{20.3}$$

An alternative, equivalent representation of this assignment of points to clusters is given by 'responsibilities', which are indicator variables $r_k^{(n)}$. In the assignment step, we set $r_k^{(n)}$ to one if mean k is the closest mean to datapoint $\mathbf{x}^{(n)}$; otherwise $r_k^{(n)}$ is zero.

$$r_k^{(n)} = \begin{cases} 1 & \text{if} \quad \hat{k}^{(n)} = k \\ 0 & \text{if} \quad \hat{k}^{(n)} \neq k. \end{cases} \tag{20.4}$$

What about ties? – We don't expect two means to be exactly the same distance from a data point, but if a tie does happen, $\hat{k}^{(n)}$ is set to the smallest of the winning $\{k\}$.

Update step. The model parameters, the means, are adjusted to match the sample means of the data points that they are responsible for.

$$\mathbf{m}^{(k)} = \frac{\displaystyle\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}} \tag{20.5}$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}. \tag{20.6}$$

What about means with no responsibilities? – If $R^{(k)} = 0$, then we leave the mean $\mathbf{m}^{(k)}$ where it is.

Repeat the assignment step and update step until the assignments do not change.

iii. Algorithm Steps[5]

- Clusters the data into k groups where k is predefined.
- Select k points at random as cluster centers.
- Assign objects to their closest cluster center according to the Euclidean distance function.
- Calculate the centroid or mean of all objects in each cluster.
- Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

3. Mean Shift Algorithm[6]

i. Description

Mean shift, a simple iterative procedure that shifts each data point to the average of data points in its neighborhood. It is shown that mean shift is a mode-seeking process on a surface constructed with a “shadow” kernel. For Gaussian kernels, mean shift is a gradient mapping. Convergence is studied for mean shift iterations. Cluster analysis is treated as a deterministic problem of finding a fixed point of mean shift that characterizes the data.

ii. Mean Shift Clustering

Let data be a finite set S embedded in the n-dimensional Euclidian space, X. Let K be a flat kernel that is the characteristic function of the A-ball in X,

$$K(x) = \begin{cases} 1 & \text{if} \|x\| \le \lambda \\ 0 & \text{if} \|x\| > \lambda \end{cases}.$$

The sample mean at $x \in X$ is

$$m(x) = \frac{\sum_{s \in S} K(s-x)s}{\sum_{s \in S} K(s-x)}.$$

The difference $m(x) - x$ is called mean shift the repeated movement of data points to the sample means is called the mean shift algorithm. In each iteration of the algorithm, $s \leftarrow m(s)$ is performed for all $s \ \in S$ simultaneously

- *A generalized procedure is as follows*

1) Randomly initialize "cluster centers," T.
2) Compute the following function on $T \times S$:

$$v_{t,s} = \begin{cases} 1, & \text{if } t = \operatorname{argmin}_T |s-t|^2 \\ 0, & \text{otherwise} \end{cases}$$

3) Update "cluster centers:"

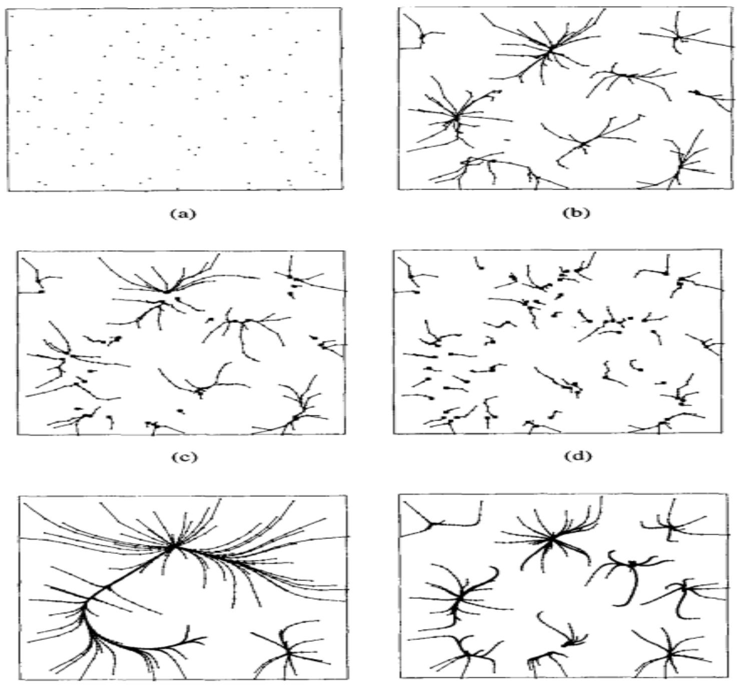
$$t \leftarrow \frac{\sum_{s \in S} v_{t,s} s}{\sum_{s \in S} v_{t,s}} \quad t \in T. \tag{11}$$

Go to 2.
Indeed, when the profile k is strictly decreasing,

$$\frac{K^{\beta}(s-t)}{\sum_{t \in T} K^{\beta}(s-t)} \rightarrow v_{t,s}, \text{ as } \beta \rightarrow \infty. \tag{12}$$

Thus, k-means clustering is the limit of the mean shift algorithm with a strictly decreasing kernel K^{β} when $\beta \rightarrow \infty$. □

iii. A figure of mean shift after x number of iterations


(a) (b) (c) (d)

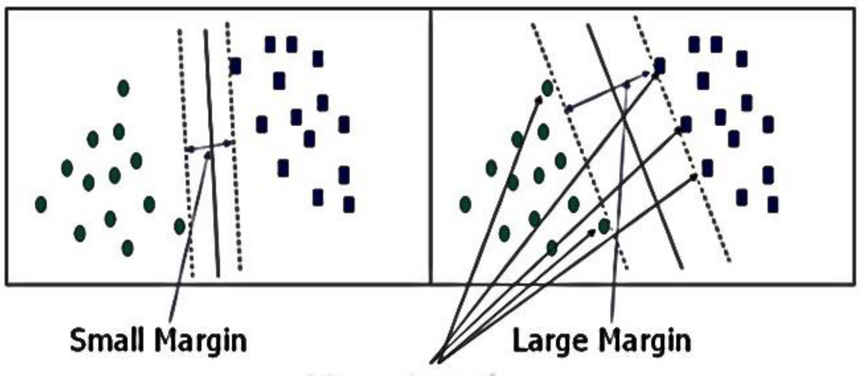
4. SVM (Support Vector Machine)

i. Description[7]

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

ii. SVM classification[8]

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



- **Large Margin Intuition**

  In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

- **Cost Function and Gradient Updates**
  In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions look as below.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 \quad = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \big(1 - y_i \langle x_i, w \rangle\big)_+ \quad = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

When there is no misclassification, i.e. our model correctly predicts the class of our data point, we only must update the gradient from the regularization parameter. Gradient update mechanism for no misclassification are as follows

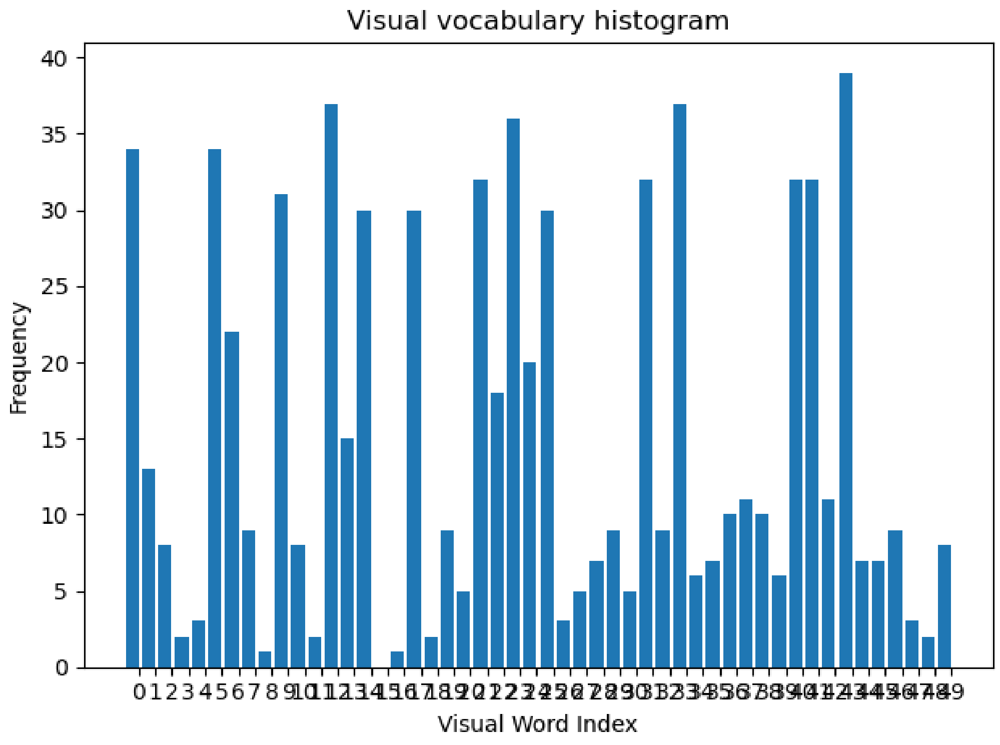
$$w = w - \alpha \cdot (2\lambda w)$$

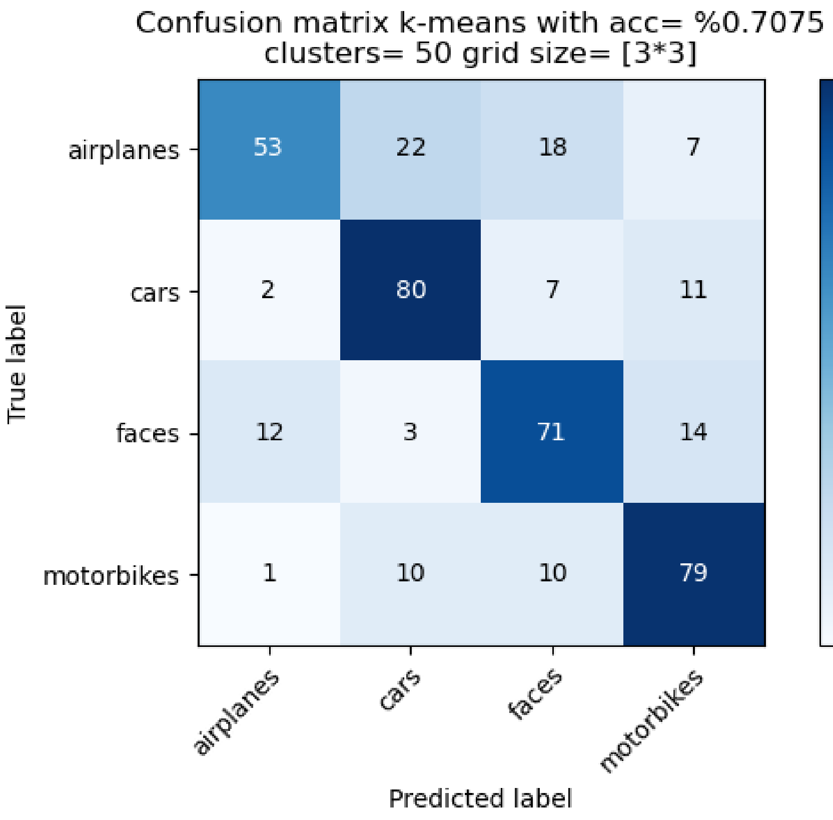
When there is a misclassification, i.e. our model makes a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update. Misclassified gradient update are as follows

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

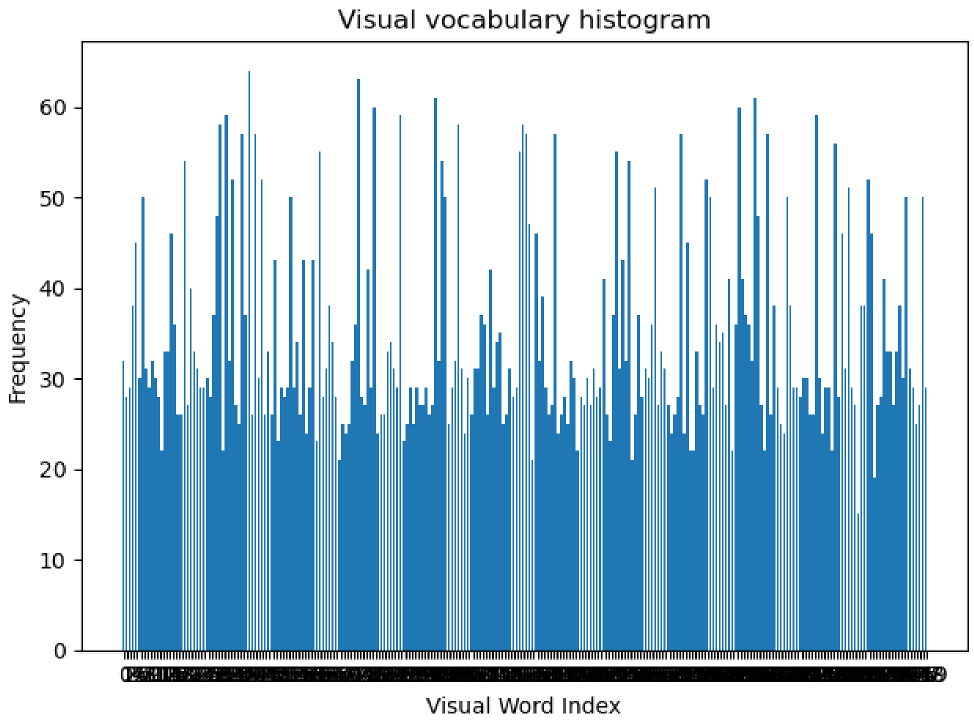
3. Tests

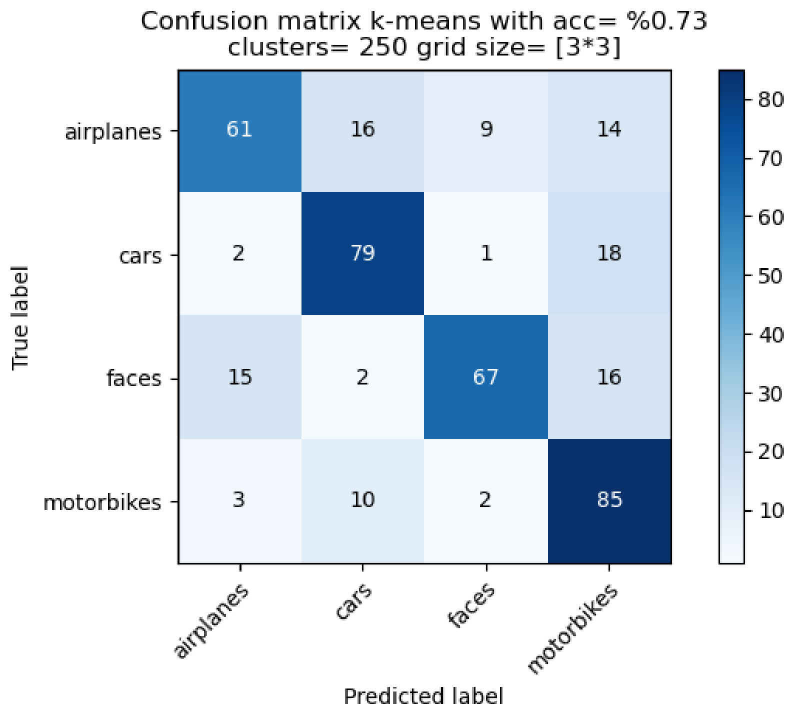
1. ***K-means Visual vocabulary Histogram k=50 grid size = [3*3]***



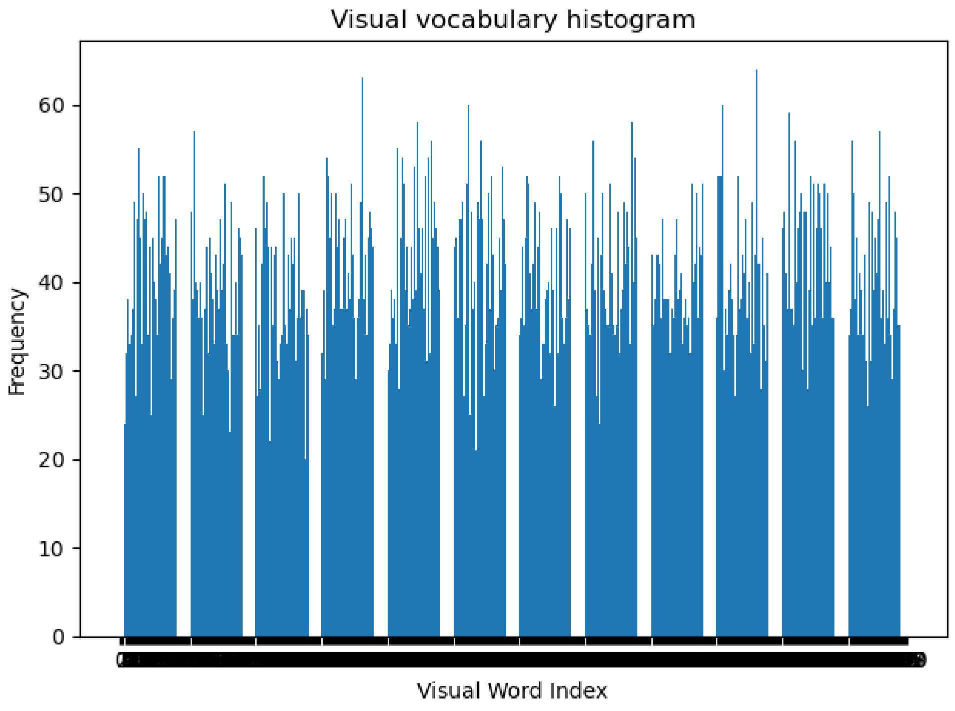


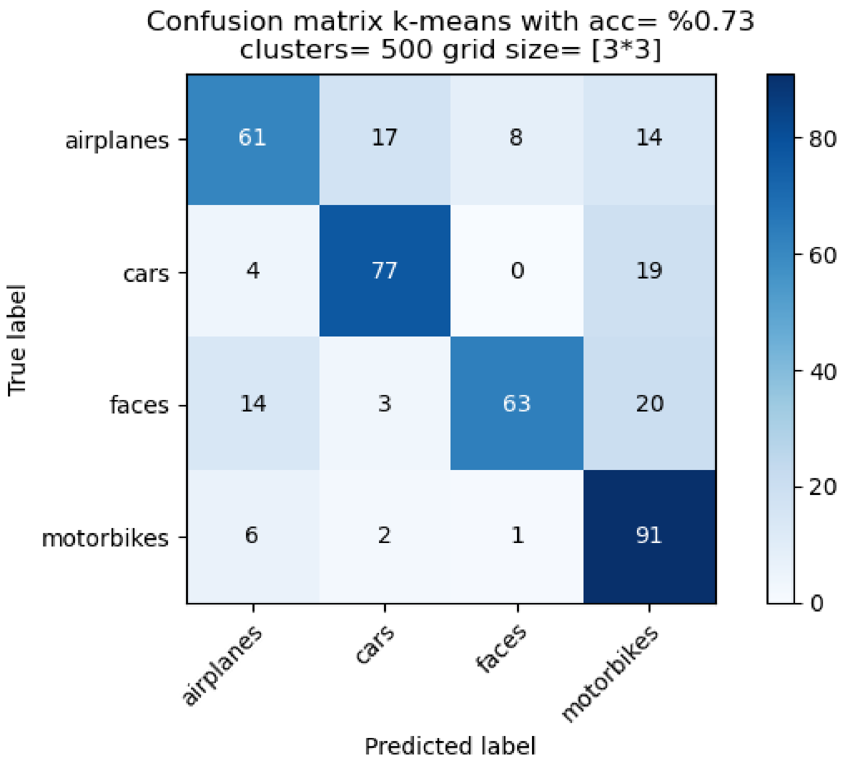
2. ***Visual vocabulary histogram k-means k = 250, grid size = [3*3]***



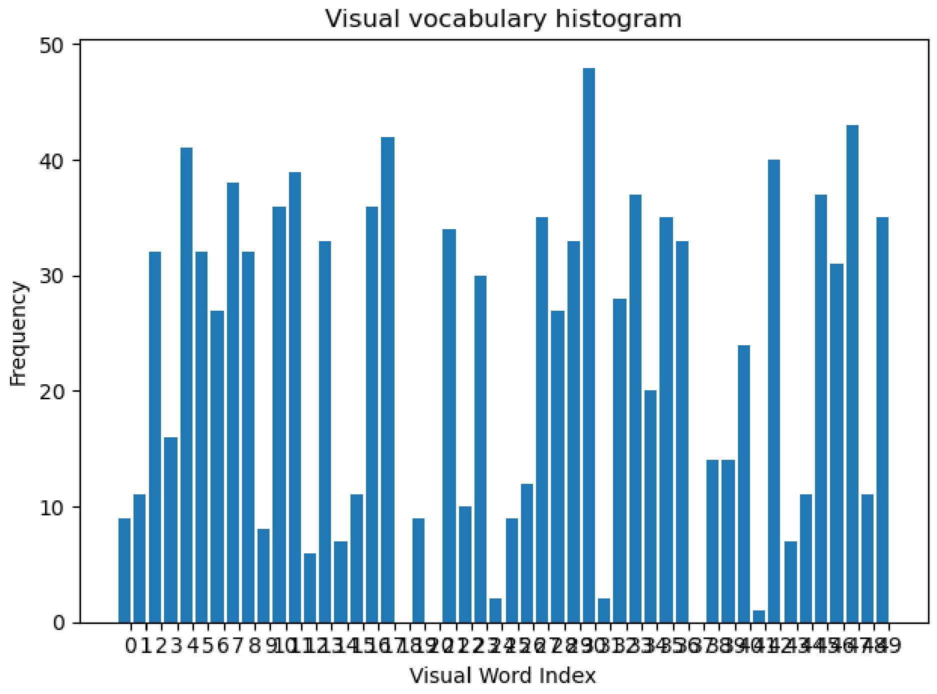


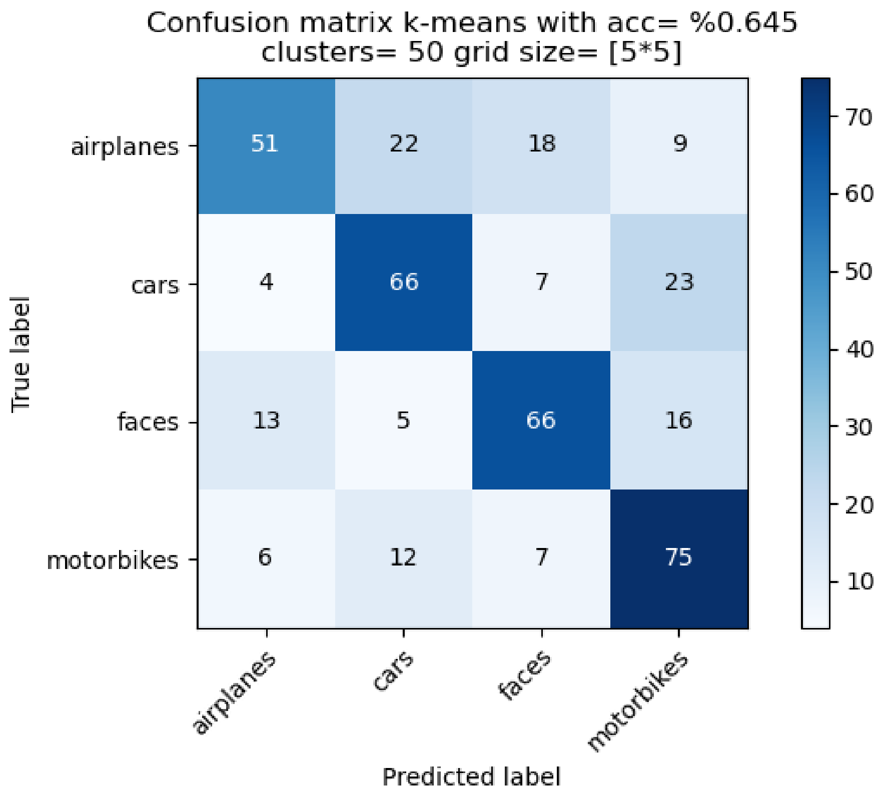
3. ***Visual vocabulary histogram k-means k = 500, grid size = [3*3]***



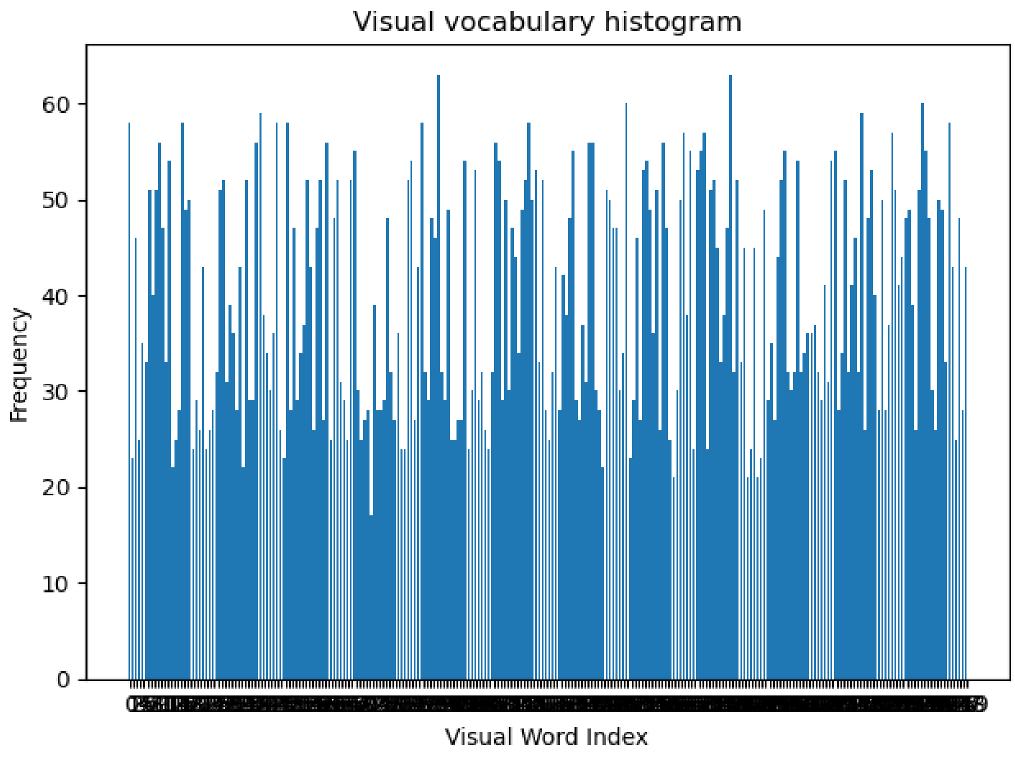


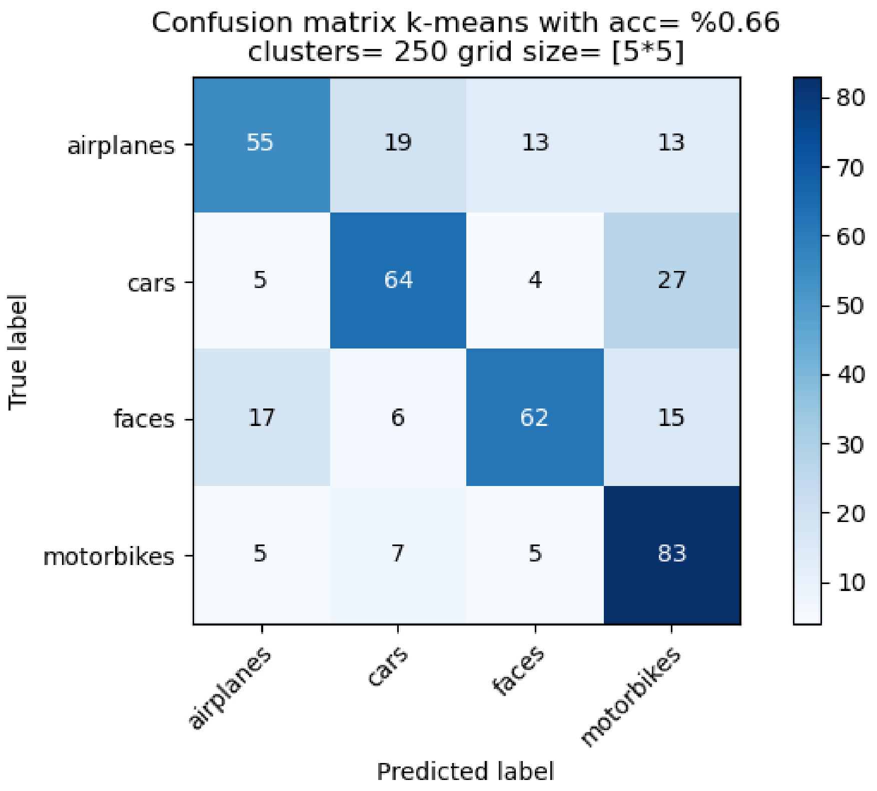
4. ***Visual Vocabulary Histogram k-means k = 50, grid size = [5*5]***



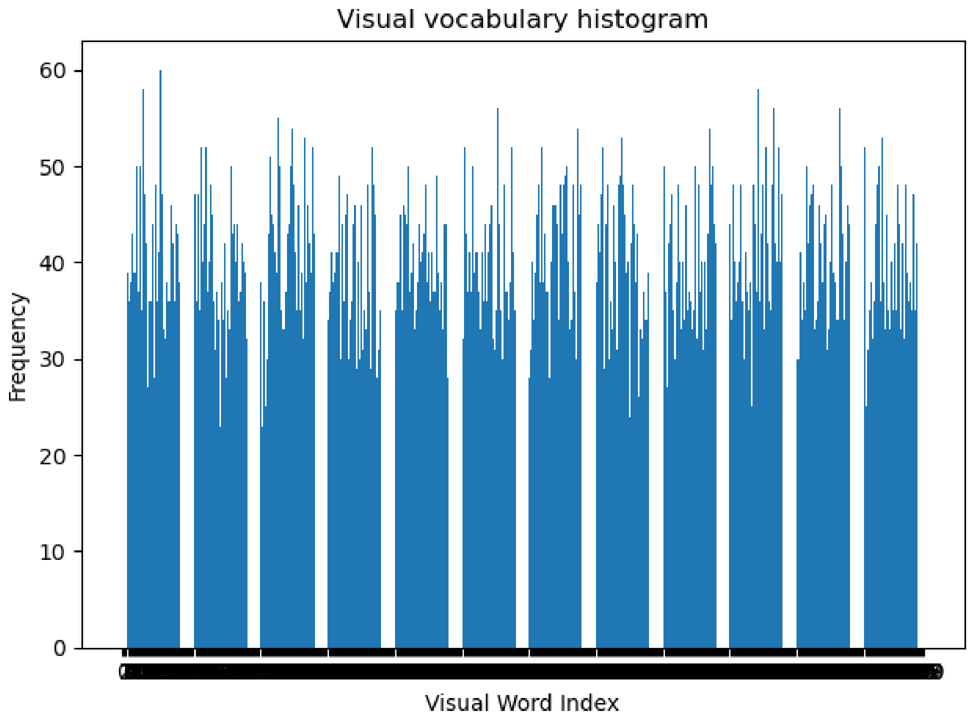


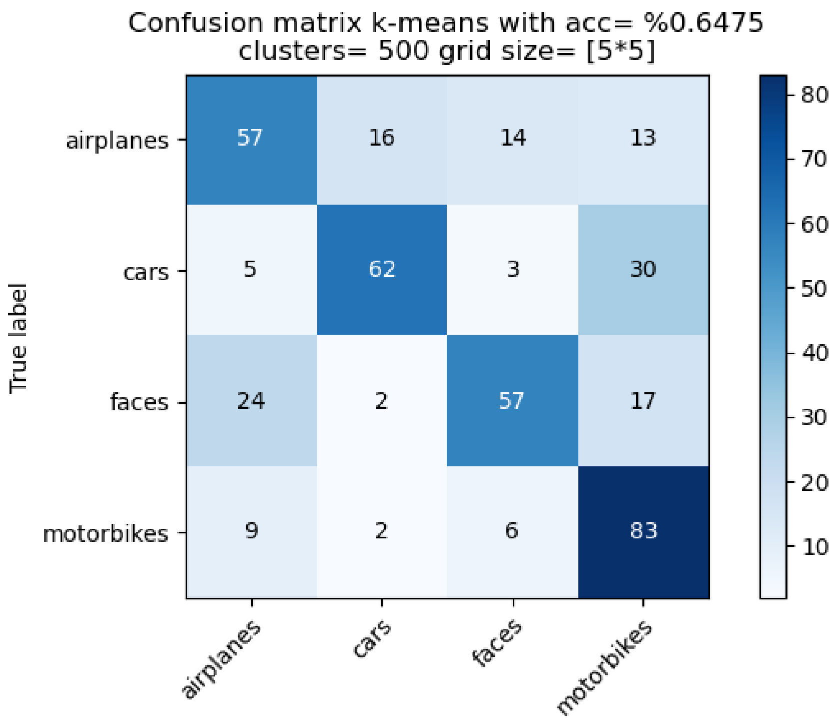
5. *Visual Vocabulary Histogram k-means k = 250, grid size = [5*5]*



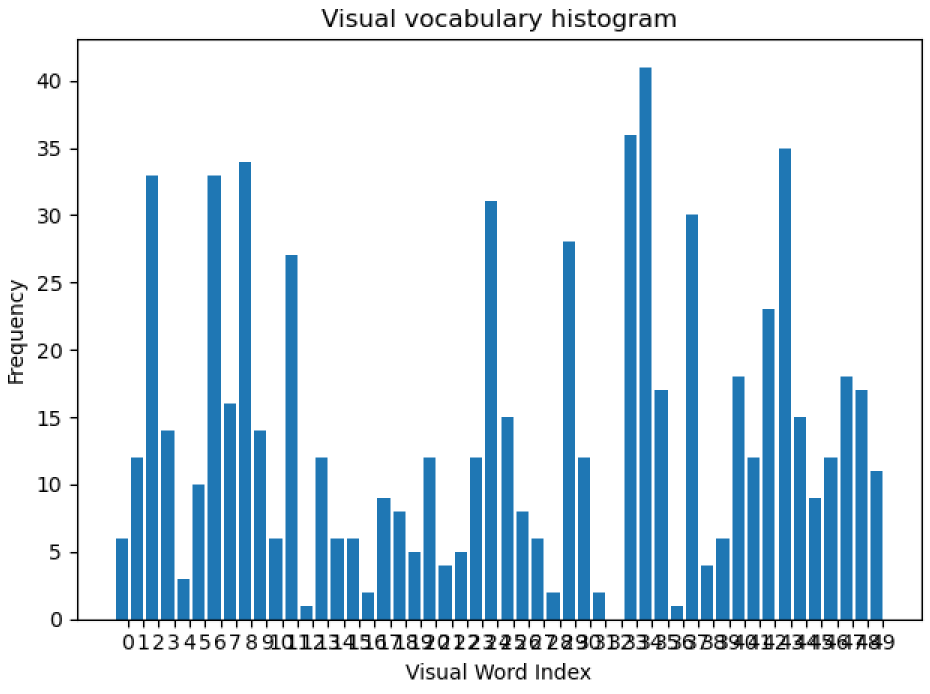


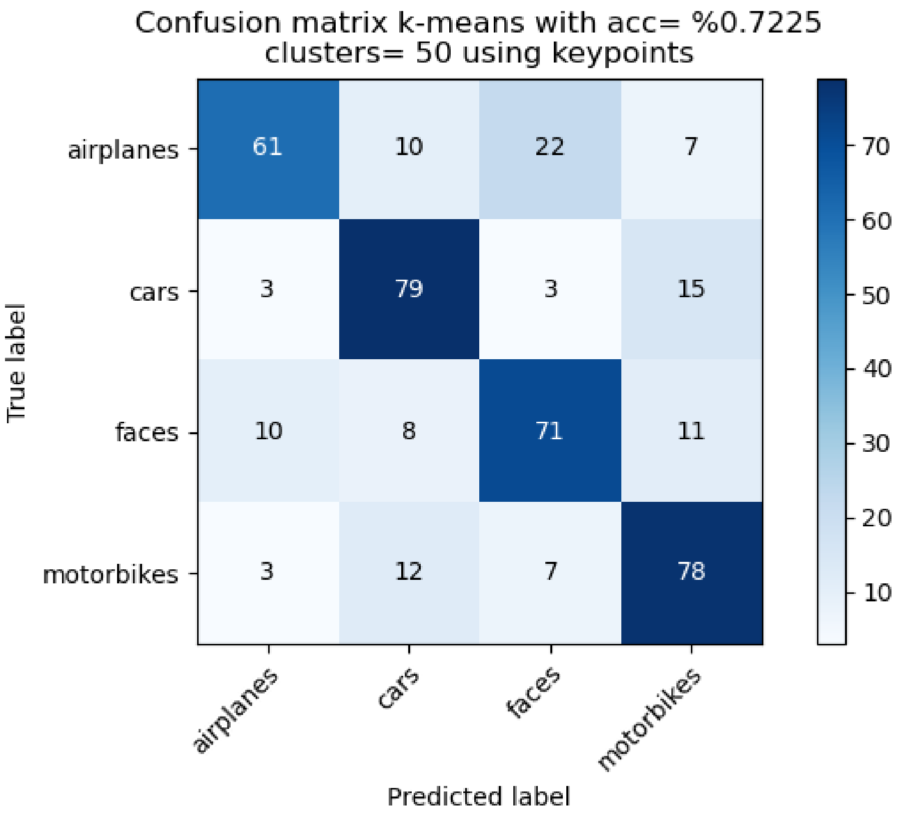
6. ***Visual Vocabulary Histogram k-means k = 500, grid size = [5*5]***



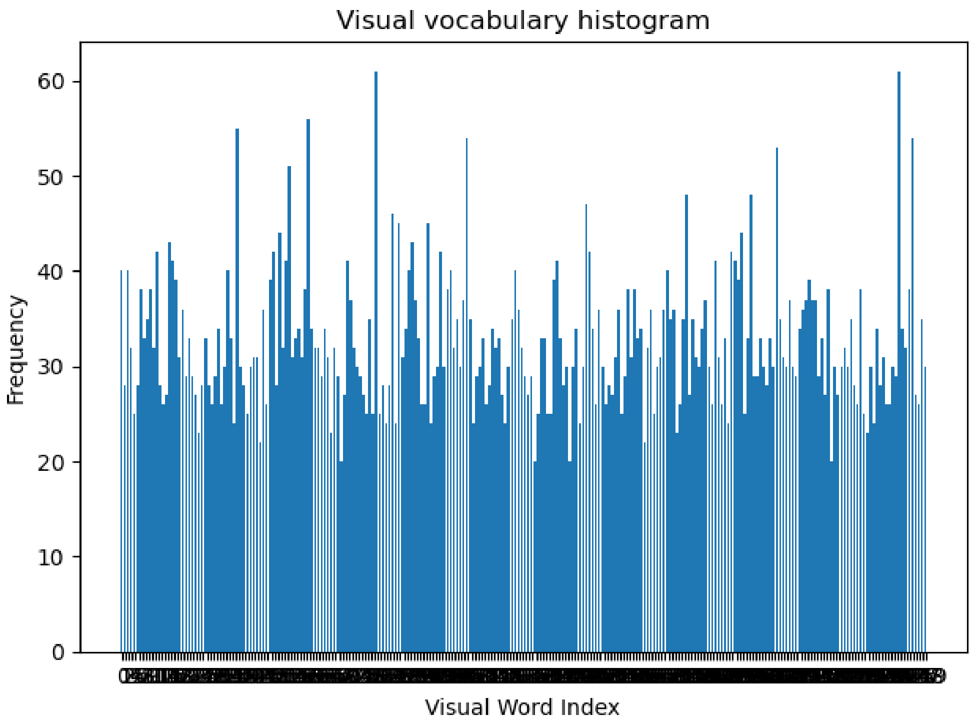


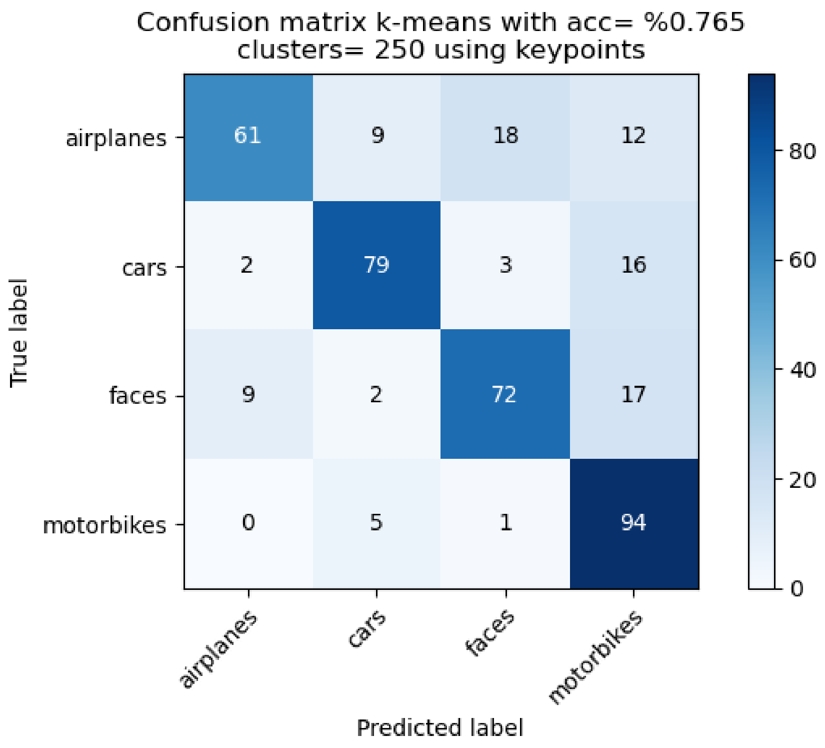
7. ***Visual Vocabulary Histogram k-means k=50 using key points***



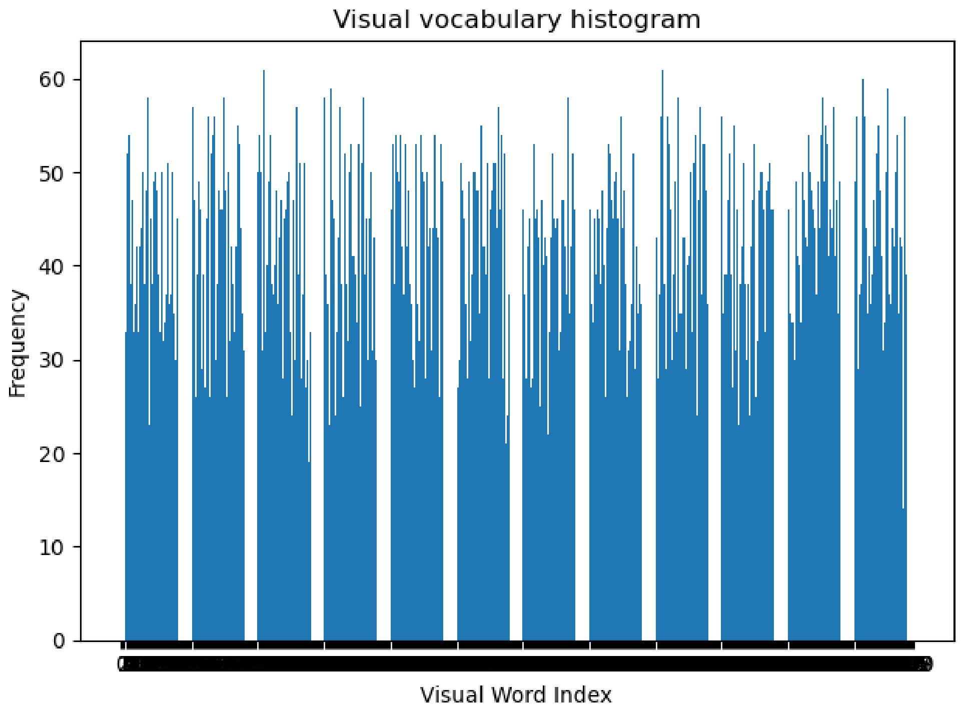


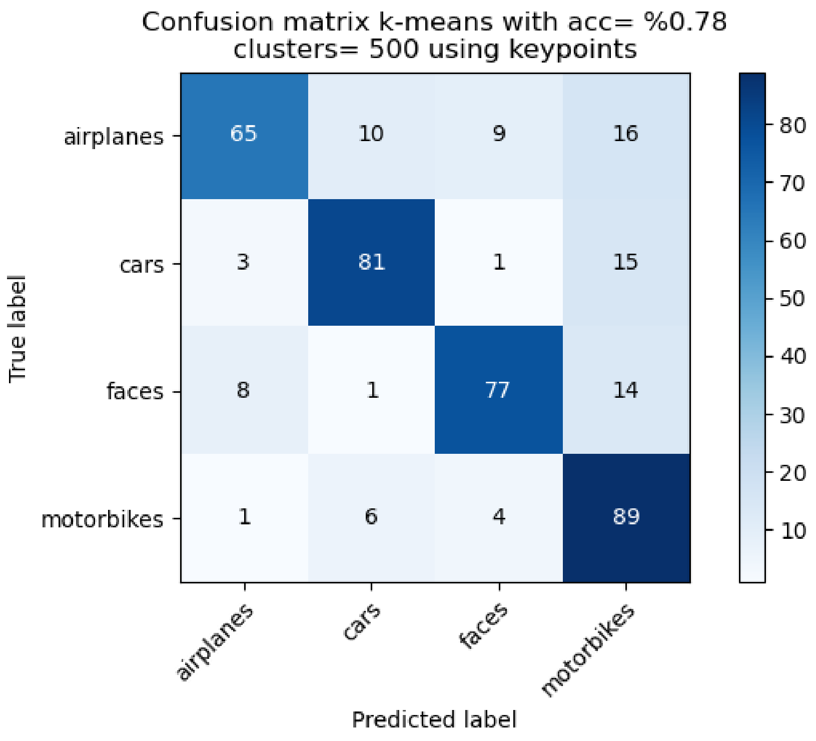
8. ***Visual vocabulary Histogram k-means k=250, used key points***

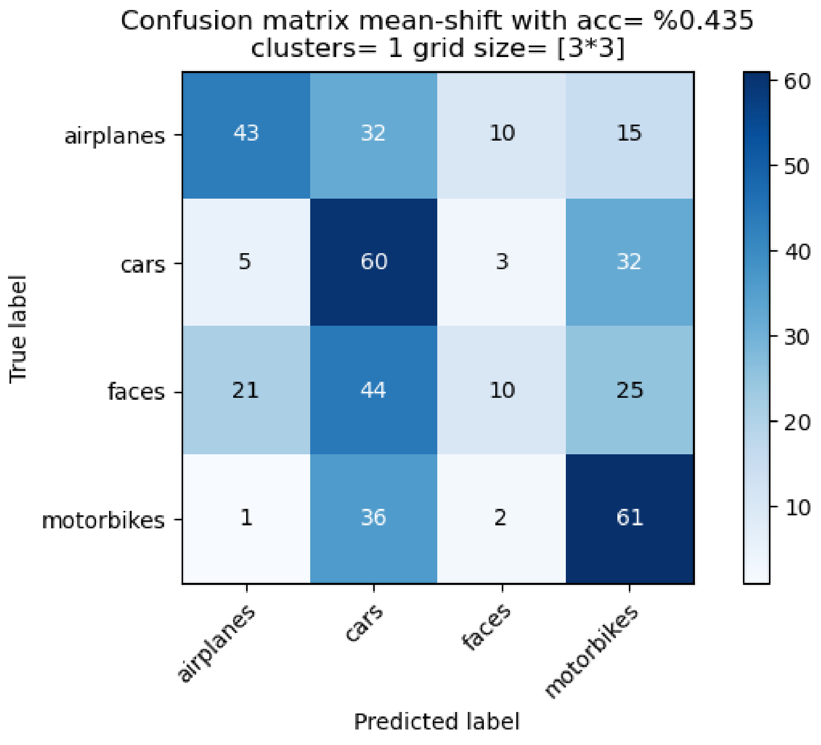


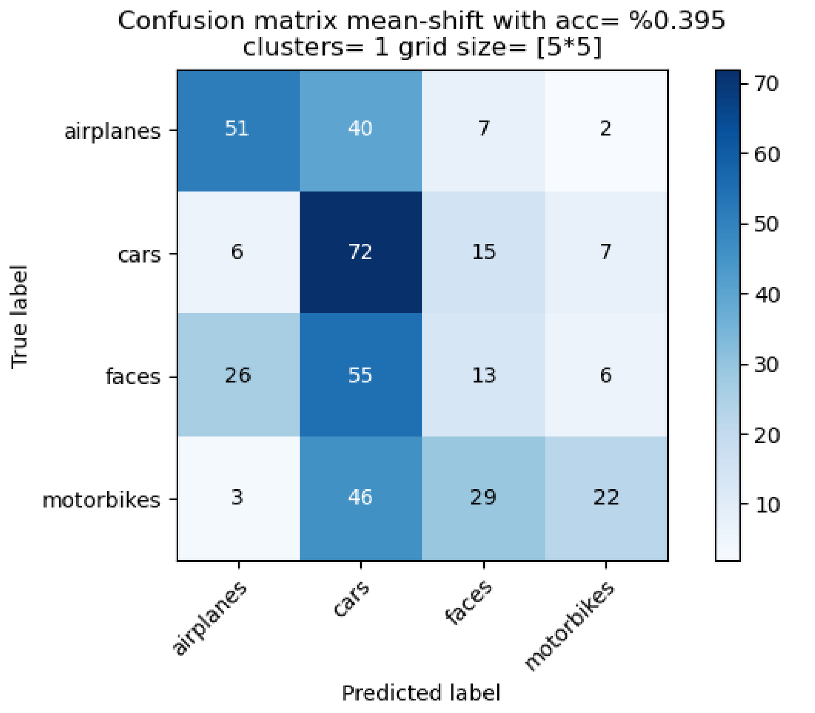


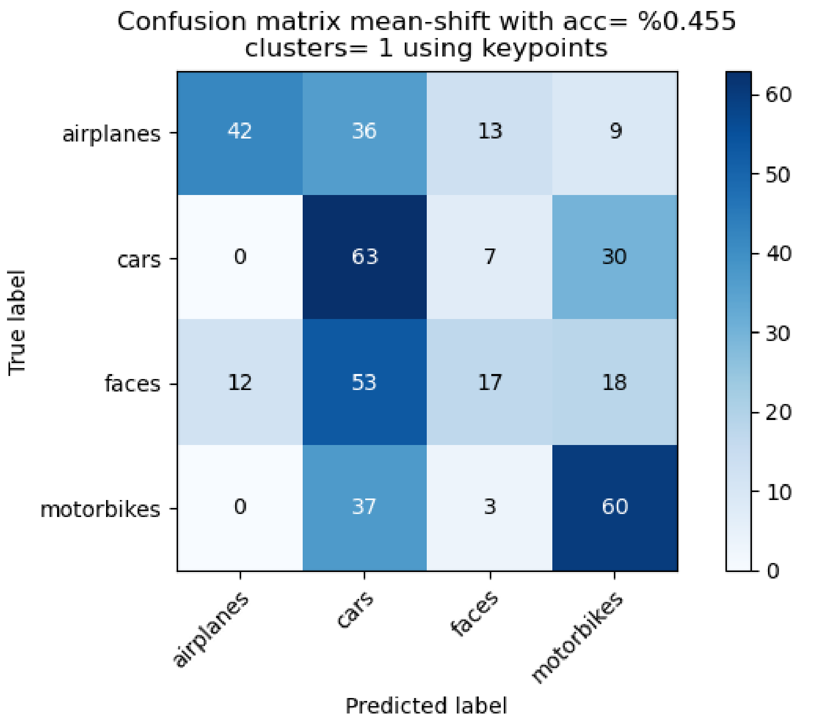
9. ***Visual vocabulary Histogram k-means k=500, used key points***




Confusion matrix k-means with acc= %0.78
clusters= 500 using keypoints
True label
Predicted label
airplanes
cars
faces
motorbikes
65
10
9
16
3
81
1
15
8
1
77
14
1
6
4
89
10
20
30
40
50
60
70
80


Confusion matrix mean-shift with acc= %0.435
clusters= 1 grid size= [3*3]
True label
Predicted label
airplanes
cars
faces
motorbikes
43
32
10
15
5
60
3
32
21
44
10
25
1
36
2
61
10
20
30
40
50
60


Confusion matrix mean-shift with acc= %0.395
clusters= 1 grid size= [5*5]
True label
Predicted label
airplanes
cars
faces
motorbikes
51
40
7
2
6
72
15
7
26
55
13
6
3
46
29
22
70
60
50
40
30
20
10


Confusion matrix mean-shift with acc= %0.455
clusters= 1 using keypoints
True label
Predicted label
airplanes
cars
faces
motorbikes
42
36
13
9
0
63
7
30
12
53
17
18
0
37
3
60
60
50
40
30
20
10
0

4. References

1. D. Lowe. Object recognition from local scale-invariant features. In IEEE International Conference on Computer Vision, test, 1999. .
2. Ives Rey Otero. Anatomy of the SIFT method. General Mathematics [math.GM]. École normale supérieure de Cachan - ENS Cachan, 2015. English. ffNNT : 2015DENS0044ff. fftel-01226489f
3. MacQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967. https://projecteuclid.org/euclid.bsmsp/1200512992
4. MacKay, David (2003). "Chapter 20. An Example Inference Task: Clustering". Information Theory, Inference and Learning Algorithms. Cambridge University Press. pp. 284–292. ISBN 978-0-521-64298-9. MR 2012999.
5. (n.d.). Retrieved June 05, 2020, from https://www.saedsayad.com/clustering_kmeans.htm
6. Cheng, Yizong (August 1995). "Mean Shift, Mode Seeking, and Clustering". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **17** (8): 790–799.