Özyeğin University

Engineering Faculty

Computer Vision (CS523)

Final Project

Problem: Image Classification with Bag of Features

Mustafa Ghanim , Department of Electrical-Electronics Engineering

INTRODUCTION

In the course final project, we discover Bag of Features (BoF) (Visual Words) based image classification. BOF can be used for any image/object classification problem, so it is generic and obtains successful validation accuracies when the used features, dataset, and classifier models are appropriate. Image classification (recognition) is widely used in many areas, like security cameras, self-driving vehicles, tumor recognition, and plate identification. In this report we will call our method as BoF or BoW (Bag of Words).

DATASET

There are four different categories in this project to be classified; 'Airplanes', 'Cars', 'Faces', and 'Motorbikes', train and validation folders are separated for each category and contain RGB-JPG images each of an average size of ~ 430 x 300. There are 160 train image (40 per category) and 400 validation (test) image (100) per category. Thus, the used dataset is not large enough for practical real-time projects, however it is sufficient for research purpose only. Since the given images' quality and resolutions are satisfying, no pre-processing operation is applied to the images before features calculations except converting RGB into gray-scale format. Gray-scale conversion is good idea when colors of the objects to be classified do not matter, like this project's case, however when classifying objects for example like 'Apple' and 'Orange', it is better to consider R-G-B channels of each image.

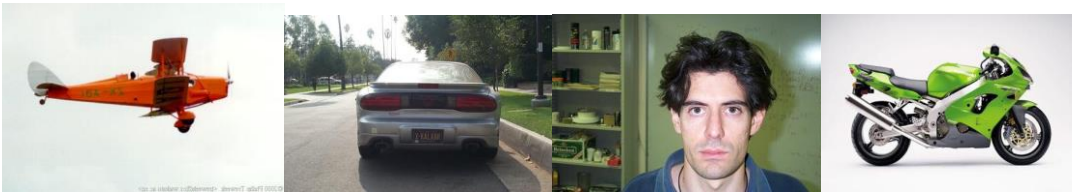

Figure 1: Examples of used train/validation images

FEATURE EXTRACTION

This step can be considered as the first step for any real classification problem, after reading the target objects and pre-processing them. We can define a feature in image processing by the visual information (it can be location, intensity, … etc.) that is considered as useful to describe the image's properties which can separate it from other type of images (classes). In this project it is required to use three type of features to be described using (SIFT):

Grid-I: consists of regular image patches, each patch has size of **32 x 32** pixels and separated by **20** pixels horizontally and vertically.

Grid-II: consists of regular image patches, each patch has size of **40 x 40** pixels and separated by **30** pixels horizontally and vertically.

Keypoints: scale-invariant features found using David Lowe's algorithm. These features are extracted by comparing each candidate image's feature with the stored corresponding features in the stored database. The similarity measure is based on Euclidean distance of the feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. Since the number of the found keypoints features in each image may be very large (depending on the image), the keypoints maximum threshold is set to be limited by **500** per image for faster computation.

Grid-I and Grid-II types of features can be called as **Dense SIFT** features as well in the literature.

Visual representation of the selected three type of features for different example images is shown:

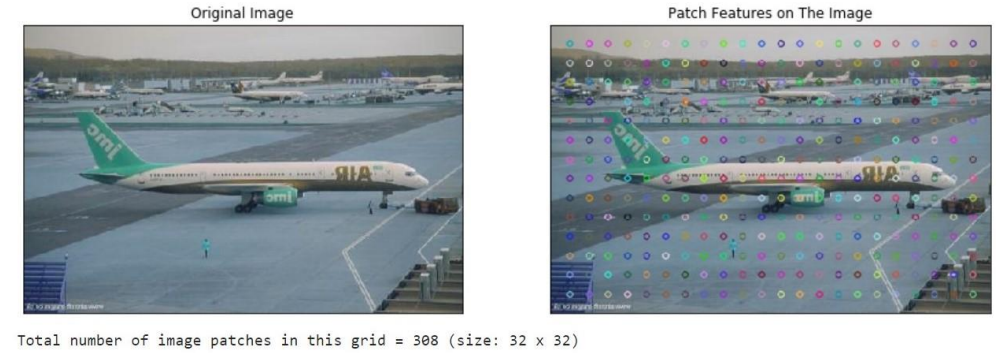

Figure 2: Grid-I type extracted features (centers of the patches)

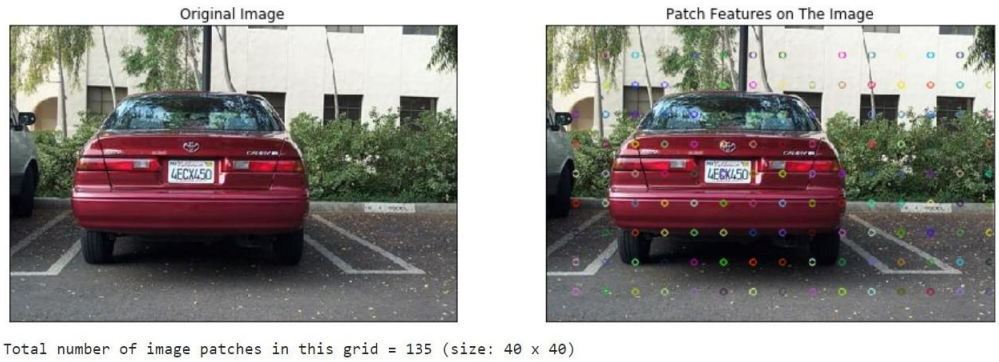

Figure 3: Grid-II type extracted features (centers of the patches)

Figure 4: Lowe's keypoints type of extracted features

SIFT DESCRIPTORS

SIFT or Scale-Invariant Feature Transform addresses the problem of matching features with scale and rotation changes. It is important to 'describe' the detected features and uniquify them against changes that any image may be exposed like; illuminance, rotation, scale, affine, and perspective transformations. Thus, the resulted classification results will be more accurate. In fact, for feature detection (keypoints localization) SIFT algorithms uses Lowe algorithm by default. However, the implemented SIFT descriptor function accepts supplied different features like grid patches as well. To create scale invariance, the interest points are scanned at a wide range of scales and the scale where the interest point features, i.e. the 2D space wise signal variances, meet certain stability criteria, is selected as the one that gets coded by the descriptor. In this way when the same process is run on a candidate interest point for recognition, it will get the same description regardless of actual scale, as the scale with the most stable presence of the interest point features will be coded. In this way we have scale invariance. In general, after finding each keypoints location (scale-invariant ones), in a **16 x 16** window the histograms of gradient directions are calculated around each keypoint. The calculated gradient directions are related to the edge orientations inside the window, and a threshold value of angle can be set the throw out the weak edges. After creating the total angle histogram of the **16 x 16** window and thresholding weak bins correspondences. Each window is again divided into **4 x 4** grid of cells and angle histogram is again calculated for each cell. Thus, for each keypoint feature an orientation descriptor (histogram) is formed in the size of **8** (orientations) x **16** (cells) = **128** dimension. Thus, the final descriptors become both scale and rotation invariant. Such properties, make SIFT to be a perfect choice in images matching (classification) problems.

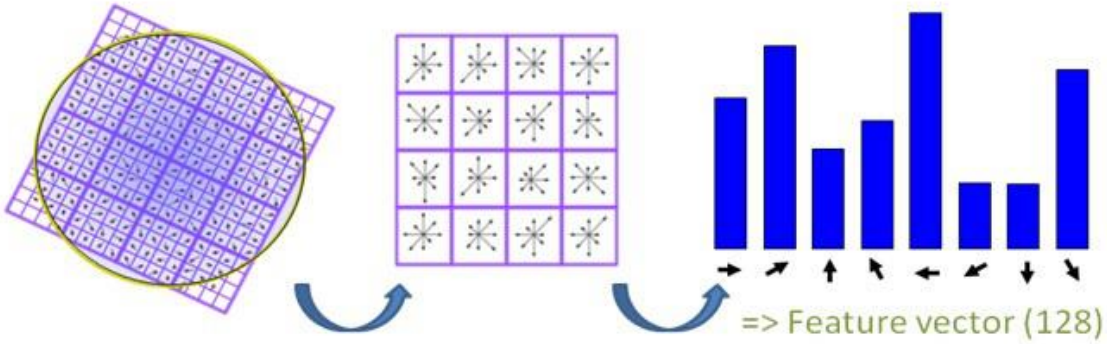

Figure 5: SIFT descriptors (orientation histograms)

```
1 features_size(train_image_paths,grid_sift= True,step = 20,size = 32)
```

```
The size of SIFT discriptors for this experiment = 45018 x 128
The average SIFT discriptors  per image = 281.36
```

Figure 6: SIFT descriptors sizes for Grid-I features

```
1 features_size(train_image_paths,grid_sift= True,step = 30,size = 40)
```

```
The size of SIFT discriptors for this experiment = 19288 x 128
The average SIFT discriptors  per image = 120.55
```

Figure 7: SIFT descriptors sizes for Grid-II features

```
features_size(train_image_paths,grid_sift= False,step = 20,size = 32)
```

```
The size of SIFT discriptors for this experiment = 65137 x 128
The average SIFT discriptors  per image = 407.11
```

Figure 8: SIFT descriptors sizes for Lowe keypoint features

Although that the largest number of SIFT descriptors for all train images belongs to Lowe keypoints based features, the first two methods (Dense SIFT based) give better accuracy results as we will see, since they are more organized, regular, and do not ignore any specific area in the image. However, Lowe keypoints can also show better results in different classification area.

BoF Dictionary

The idea behind BoF or BoW dictionary formulation is very similar to language dictionary as concept, but in computer vision we create a dictionary of visual words; SIFT descriptors or keypoints. The input to this dictionary, should be all the train images, and it becomes more generic and precise as we increase the contained visual words. Nevertheless, too large dictionary size can lead to overfitting problems in feature quantization step.

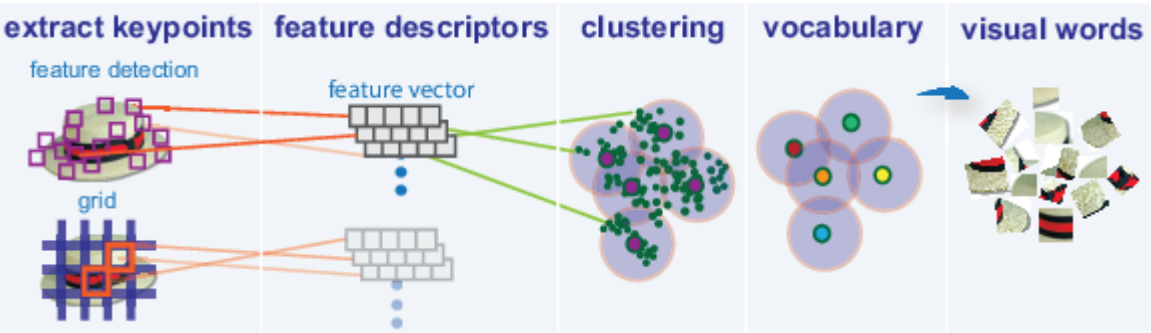

Figure 9: Bag of Visual Words dictionary system

Dictionary formulation steps can be summarized by the figure above. In the project as it is mentioned; 3 different types of keypoints (features) are used, one type of descriptors (SIFT), and two type of clustering algorithms (Kmeans and Means-Shift) are used. Clustering found descriptors is done to extract only the most important and separated visual words of the dictionary, i.e. the centroid descriptors of each cluster. Thus, feature quantization and histogram calculation of each train and test image becomes easier in terms of assigning input image descriptors to the 'Nearest' frequency terms in the codebook (dictionary).

K-MEANS CLUSTERING

K- Means is a non- hierarchical clustering algorithm. It is an iterative algorithm which tries to partition the dataset into a pre-defined number of non-overlapping clusters(k) by minimizing the sum of squared distances from each data point to the cluster centroid (Within sum of squared distances- WSS). The algorithm flow can be explained as:

1. Pre-define the no. of clusters (K)
2. Initialize cluster centroids by randomly selecting K data points
3. Calculate the WSS distances between each data point to all the cluster centroids
4. Assign each data point to the closest centers

Steps 3 and 4 will be iterated until there is no reassignment of data points is required.

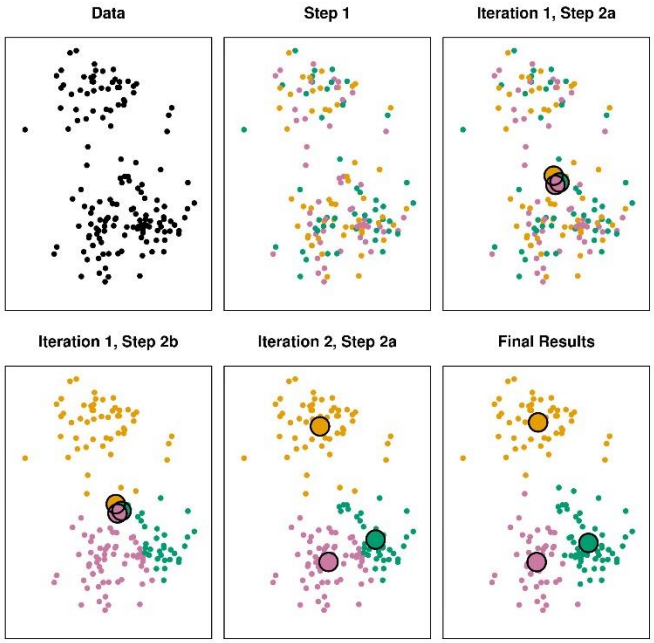

Figure 10: K-means clustering example

MEAN-SHIFT CLUSTERING

Mean shift considers feature space as an empirical Probability Density Function (PDF). If the input is a set of points then Mean shift considers them as sampled from the underlying probability density function. If dense regions (or clusters) are present in the feature space, then they correspond to the mode (or local maxima) of the probability density function. We can also identify clusters associated with the given mode using Mean Shift.

For each data point, Mean shift associates it with the nearby peak of the dataset's Probability Density Function. For each data point, Mean Shift defines a window around it and computes the mean of the data point. Then it shifts the center of the window to the mean and repeats the algorithm till it converges. After each iteration, we can consider that the window shifts to a denser region of the dataset. The algorithm flow can be explained as:

1. Fix a window around each data point.
2. Compute the mean of data within the window.
3. Shift the window to the mean and repeat till convergence.

As explained, K-means requires the number of clusteres as its input while Mean-Shift can find the required K size of clusters by itself. In this project, experiments of using K-means

only, Means-Shift only, and Mean-Shift + K-means (K found Mean-Shift and clustering by K-means) are done.

It is important to state that when using K-means for clustering, SIFT descriptors array is enough to be provided directly to the algorithm, however when using Mean-Shift for clustering we need to supply MS input with the locations of the keypoint features since Mean-Shift is related to position changes in the dataset (like the object tracking example in the lectures). Clustering SIFT descriptors directly with MS gives incorrect results. Since in somehow we need to form a new centriod SIFT discriptors of the new clusters, we calculate the arithmetic mean of SIFT descriptors that correspond to each found label of each cluster.

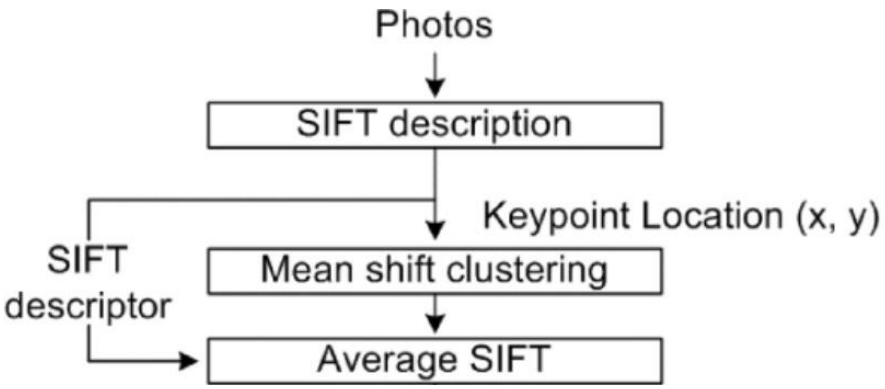

Figure 11: Clustering operations with Means-Shift

One important parameter of the built-in Sklearn function of MS is the bandwidth; distance/size scale of the kernel function, i.e. what the size of the "window" is across which we calculate the mean. ***sklearn.cluster.estimate_bandwidth*** function tries to estimate the required bandwidth for the given dataset, however when it is tested it gives very large values for bandwidth leading to bad and long calculations for this dataset. Thus, a bandwidth scale of '15' is found to be giving very proper results by Try-and-Error directly.

HISTOGRAM CALCULATION AND FEATURE QUANTIZATION

After calculation each BoF for each expierment, their output files are saved using 'Pickle' Python Object Serilization in order to be used later. After that we again load for corresponding dictionary file and calculate the SIFT descriptors of each training and test image (separately). Note that we use the same experiment parameters of feature type in this step too to avoid any possible mismatch and calculation irrationality. This step can be considered as the final 'Feature Extraction' step since its output is supplied into the SVM classifier. We calculate the visual words for each train/validation image by determining the closest visual words from the given codebook using 'Nearest Neighbour' quantization. After

that, the histograms (frequency) of the found visual words are calculated. Thus, each training and validation image will be featured with its quanztizied and different frequency terms (histograms of visual words). Finally, we have the labels of train/validation images as the given category names in the begining, moreover, their quantized features are stored in two different arrays as well.

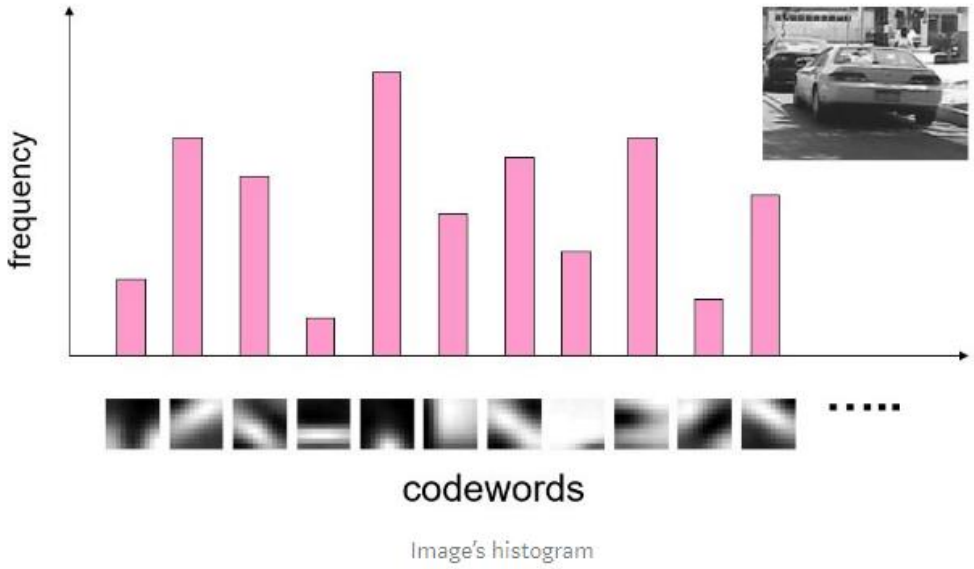

Figure 12: Quantized histogram frequencies of a car image

As it can be seen in figure 12, the car image can be represented in such frequency (counts) of the found visual words from the dictionary. Each different frequency term corresponds to one the clusters found using K-means or Mean-Shift. Such important features work very successfully with Linear SVC classifier (one type of SVM). Moreover, as an extra (not required) step that can be applied on the found frequencies of each image, is applying what is called tf–idf or TFIDF, short for term frequency–inverse document frequency, which is a numerical statistic that is intended to reflect how important a word (or visual word) is to a document (train/validation image) in a collection (codebook). It is beyond the topic of this project; however, its function is provided as extra optimization step (not used) but can be useful in future works. Its effect may show up when some visual words are more important than other visual words in specific cases.

SUPPORT VECTOR MACHINE CLASSIFICATION

Due to its simplicity and compatibility with our classification problem; we use Linear SVC (Support Vector Classifier) and fit the training data into it, returning a "best fit" hyperplane that divides, or categorizes, the data. From there, after getting the hyperplane, we can then feed the validation quantized features to the classifier to see what the "predicted" class is. The hyperplane maximizes the margin between the two classes or (multiple classes). The vectors (cases) that define the hyperplane are the support vectors. Moreover, Linear SVC handles the One – vs – Rest classification scheme.

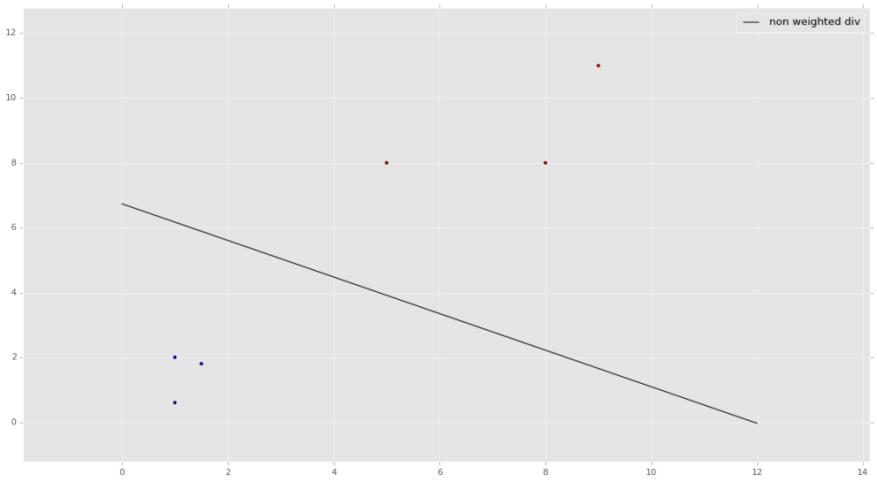

Figure 13: Separation of 2-D features using SVC

In our SVC (or SVM) model inputs we change default inputs as follows:

1. random_state: passed as ‘0’ for reproducible output across multiple function calls
2. tol: tolerance for stopping criteria, passed as ‘1e-4’
3. C: regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. Passed as ‘7’ to reduce the number of miss-classifications as much as possible
4. kernel: passed as ‘linear’

The predicted class of each test (validation) image is found and compared with the actual test label, then the confusion matrix, validation accuracy (mean of the diagonal of the CM), and FP/FN values are calculated. Confusion matrix can be found using ***sklearn.metrics. confusion_matrix*** as well as the provided manually implemented function in 'main.py'.

IMAGE CLASSIFICAITON EXPERIMENTS AND RESULTS

As required in this project, the following experiments are performed for image classifications.

Exp. No	Feature Extraction	Feature Descriptor	Dictionary Computation	Feature Quantization and Histogram Calculation	Classifier	Validation Accuracy
1	Grid-I (32 x 32 patch size)	SIFT	Mean-Shift clustering, found k = 302	Nearest Neighbor	SVM (linear kernel)	**96.00%**
2	Grid-I (32 x 32 patch size)	SIFT	K-means: k = 50	Nearest Neighbor	SVM (linear kernel)	**97.25%**
3	Grid-I (32 x 32 patch size)	SIFT	K-means: k = 250	Nearest Neighbor	SVM (linear kernel)	**97.50%**
4	Grid-I (32 x 32 patch size)	SIFT	K-means: k = 500	Nearest Neighbor	SVM (linear kernel)	**98.00%**
5	Grid-I (32 x 32 patch size)	SIFT	K-means: k = 302 (k is found by Mean-Shift)	Nearest Neighbor	SVM (linear kernel)	**97.50%**
6	Grid-II (40 x 40 patch size)	SIFT	Mean-Shift clustering, found k = 171	Nearest Neighbor	SVM (linear kernel)	**93.75%**
7	Grid-II (40 x 40 patch size)	SIFT	K-means: k = 50	Nearest Neighbor	SVM (linear kernel)	**95.75%**
8	Grid-II (40 x 40 patch size)	SIFT	K-means: k = 250	Nearest Neighbor	SVM (linear kernel)	**98.00%**
9	Grid-II (40 x 40 patch size)	SIFT	K-means: k = 500	Nearest Neighbor	SVM (linear kernel)	**98.25%**
10	Grid-II (40 x 40 patch size)	SIFT	K-means: k = 171 (k is found by Mean-Shift)	Nearest Neighbor	SVM (linear kernel)	**97.00%**
11	Scale-invariant keypoints	SIFT	Mean-Shift clustering, found k = 138	Nearest Neighbor	SVM (linear kernel)	**64.75%**
12	Scale-invariant keypoints	SIFT	K-means: k = 50	Nearest Neighbor	SVM (linear kernel)	**63.00%**
13	Scale-invariant keypoints	SIFT	K-means: k = 250	Nearest Neighbor	SVM (linear kernel)	**72.75%**
14	Scale-invariant keypoints	SIFT	K-means: k = 500	Nearest Neighbor	SVM (linear kernel)	**74.50%**
15	Scale-invariant keypoints	SIFT	K-means: k = 138 (k is found by Mean-Shift)	Nearest Neighbor	SVM (linear kernel)	**70.00%**

Table 1: Performed experiments details and results

The best accuracy is done with experiment-9 and the worst accuracy is obtained from experiment-11. The average validation accuracy for all experiments is **87.6%** which is really powerful and good result due to different scales and features being used. Moreover, as explained before; 'Dense' SIFT based classifications show much better results than scale-invariant keypoints based experiments. However, even if they use a smaller number of features per image (depending on the selected grid size), they need more computation power due to sliding window operations for each image. From the results we can see clearly the effect of using larger clusters number (k) leading to more precise classifications, but again this requires more calculations for dictionary calculation step. In addition, very large (k) may introduce negatively unexpected results.

Confusion matrix visual representation, numeric values, and false negative/positive values for each experiment are shown in the following figures.

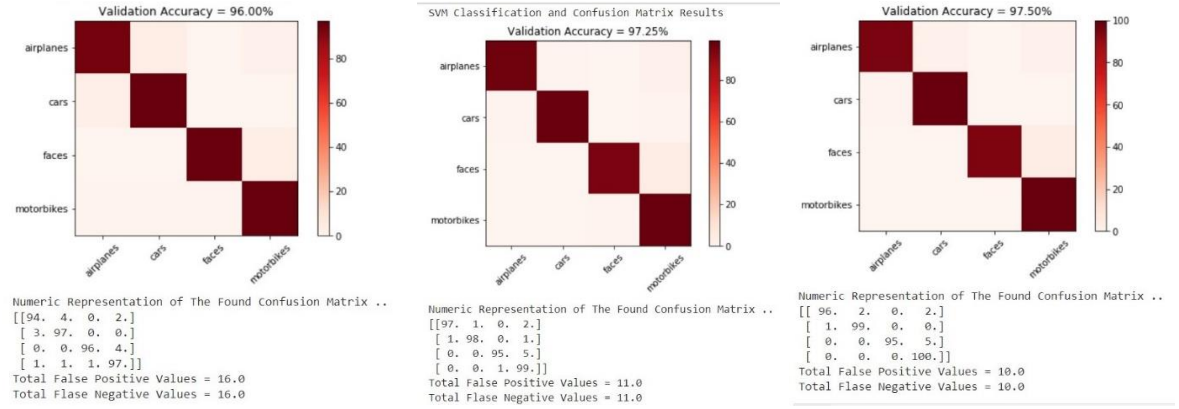

Figure 14: Results of experiments 1-2, and 3 (from left to right respectively)

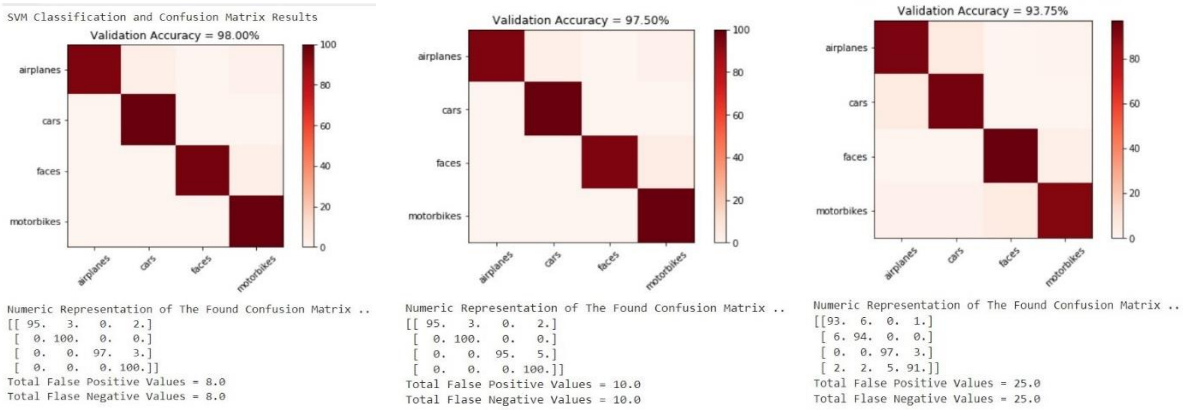

Figure 15: Results of experiments 4-5, and 6 (from left to right respectively)

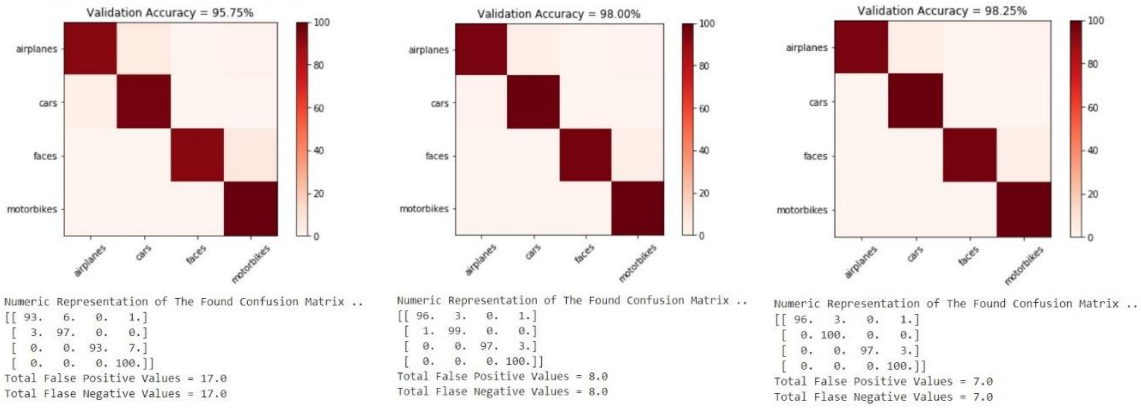

Figure 16: Results of experiments 7-8, and 9 (from left to right respectively)

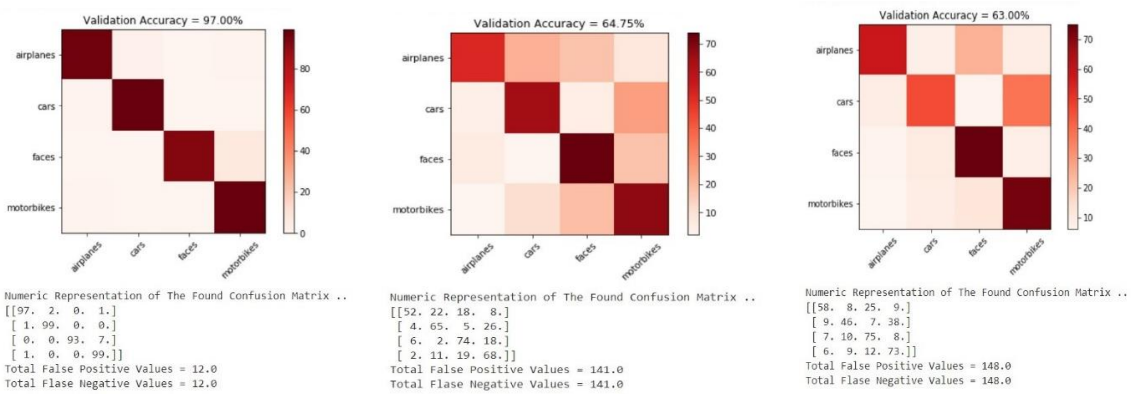

Figure 17: Results of experiments 10-11, and 12 (from left to right respectively)

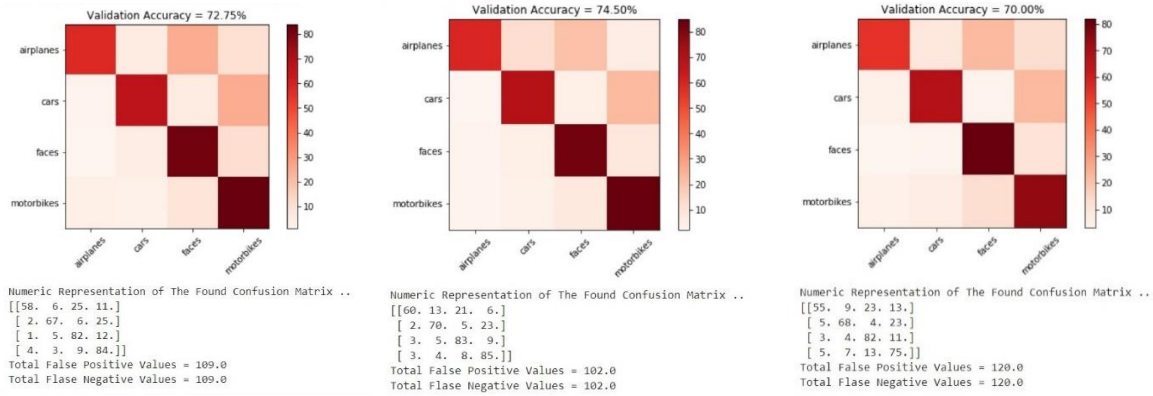

Figure 18: Results of experiments 13-14, and 15 (from left to right respectively)

Class	Average Validation Accuracy
Airplanes	**82.33%**
Cars	**86.6%**
Faces	**90.07%**
Motorbikes	**91.4%**

Table 2: Average validation accuracy per category

Figure 19: Samples of validation images with their SVM-predicted results and actual labels

REFERENCES

1. Ozyegin University, Furkan Kirac – CS523 Lecture Notes
2. UCF - Mubarak Shah - Computer Vision Video Lectures (Bag – of- Features)
3. What is the k-Means algorithm and how does it work? – Quora
4. What is the mean-shift algorithm? – Quora
5. J. Keum, H. Lee and M. Hagiwara, "Mean shift-based SIFT keypoint filtering for region-of-interest determination," The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems, Kobe, 2012, pp. 266-271, doi: 10.1109/SCIS-ISIS.2012.6505144.
6. Displaying examples/samples of Visual word from Bag of Visual Words in python OpenCV, Stack Overflow