Assignment 3 - Local Descriptor Based Image Classification

Introduction

In this assignment, we worked as a group to classify a set of images using various methods and experimented with combinations of these methods. The image prediction pipeline consists of 4 steps:

- Feature Extraction
- Finding Dictionary Centers
- Feature Quantization
- Classification

We have implemented a number of methods to achieve these goals. We, then, combined these methods to create full pipelines that predict image tags. The “Caltech20” dataset given with the assignment is used for training and testing.

Preprocessing

As it has been suggested, we have resized the images to lower the computational cost.

Feature Extraction

ORB (OpenCV implementation)

Files: feature_extraction/orb.py
Author: Mahmut Karaca

I have written a wrapper for the ORB feature extraction method implemented in the OpenCV library. As the homework requires, I have used the Python bindings for the library. The wrapper is simply a class with a single static method which takes image path as parameter and returns the key points and descriptors as a tuple.

SIFT (OpenCV implementation)

Files: feature_extraction/sift.py
Author: İpek Erdoğan

I have used the OpenCV implementation of Scale-Invariant Feature Transform (SIFT) in my pipeline to extract the descriptors from the images. SIFT extracts different numbers of descriptors from each image, which are 128-dimensional feature vectors. I restricted the number of descriptors per image as 100, to lower the computational cost. It gives 1611904 descriptors from the Caltech20 training dataset, if I don’t restrict the number.

HU MOMENTS DESCRIPTOR (Own Implementation)

Files: hu.py
Author: M.Yasin ADIYAMAN

I have used Hu moments to describe image patches. There are 7 Hu moments for each image patch which is resized to (50,50). Since Hu moments are applicable to silhouette images, I generated mask images first and then calculated Hu moments for mask(binary) images. For each image patch, I firstly convert color space to HSV to extract hue information. Then I convolve upon hue dimension with different padding and stride values to generate 19 masks. Therefore, I could reach 7*19=126 dimension descriptor vector for a patch. I added 2 extra elements (0) to the vector and generated a 128-dimension descriptor vector. I repeated this process for different 135 patches which resulted from 2D convolution over image with different padding (this time stride and padding values are equal). Then, for each image the descriptor object generates a 135x128 descriptor.

Finding Dictionary Centers

K-Means (Own Implementation)

Files: clustering/kmeans.py
Author: İpek Erdoğan

To determine the codewords (dictionary centers), we needed a clustering algorithm. By processing all the descriptors and extracting some center values (vectors) from them, we create some benchmarks for our feature vectors. So I implemented the K-Means algorithm. It starts with random initial K centers and iteratively updates the center values, with the average values of cluster members. “Cluster membership” determined according to the euclidean distance. A data vector belongs to the cluster whose center is closest (minimum euclidean distance) to the data vector. For this problem, the important output of K-Means function is the list of centroids. We give the centers list to Mahmut’s Bag of Visual Words implementation, and this center values will be the vocabulary.

Here, while determining the “K”, it’s important to understand its meaning. This number should at least be equal to the class number. But this is also not enough. Feature vectors should represent their class’ specific features if it’s possible to be distinguished from other class’ feature vectors. If we keep this number too low (i.e. 20), our feature vectors will be too generalized for being classified correctly and we can not expect a good accuracy from a classifier which we fed with these feature vectors.

Yet, if we keep this number too high (i.e. 1000), our feature vectors will be too specific for being classified correctly. This situation would cause overfitting. Also, we should consider the computational cost. In the light of these constraints, we did our experiments with different number of K’s to see these effects on the results.

Feature Quantization

Bag of Visual Words (Own Implementation)

Files: feature_quantization/bag_of_visual_words.py
Author: Mahmut Karaca

Bag of Visual Words were implemented as a single function. It simply generates a histogram using the image features and dictionary centers as input. Euclidean distance metric is used for calculating the nearest center.

Classification

Random Forest (Sklearn implementation)

Author: İpek Erdoğan

As a classifier, I decided to use Random Forest. Random Forest classifier consists of different Decision Trees. It classifies inputs by taking the average of the results of these decision trees or deciding on the dominant choice. Normally, with a quick research, you can see there are more examples which use SIFT with SVM than the ones which use with Random Forest. But there are some works [1][2] on comparison between the Random Forest and SVM, not only about accuracy but also about computation cost, which took my attention. I wanted to see the results with my experiments. That's why I chose Random Forest.

Evaluation

We have implemented different pipelines individually. In this part of the report, we will first look at the results of each pipeline separately and at the end, we will compare the average f1 scores of the models. We ran all of the pipelines for different number of clusters (K) which are 20,50,100.

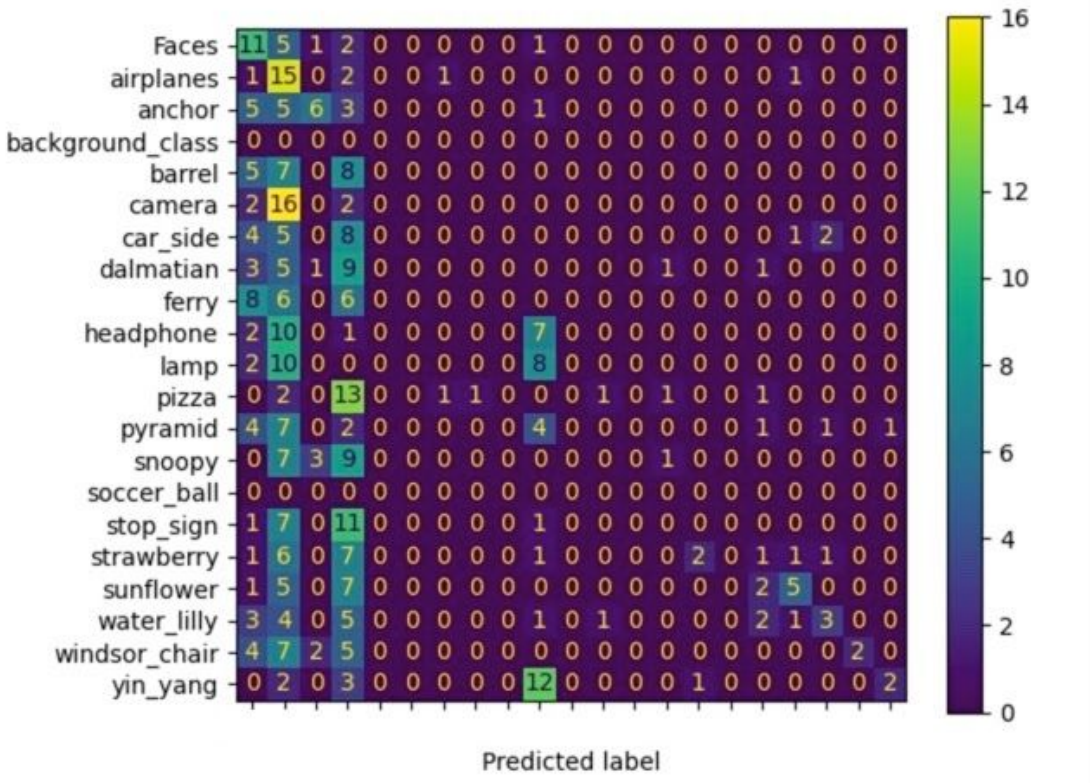
1.SIFT-K-Means (Ipek's Implementation) - Bag of Words(Mahmut's Implementation) - Random Forest

Author: İpek Erdoğan

K=20

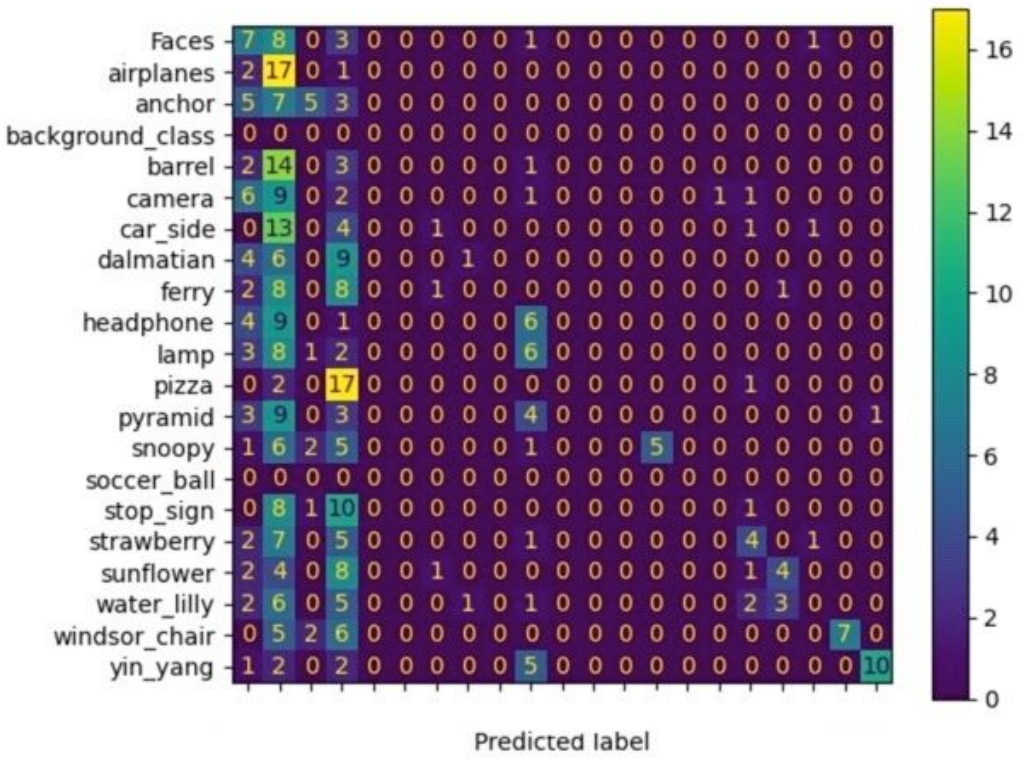
Class	Precision	Recall	F1 Score
Faces	0.19	0.55	0.29
airplanes	0.11	0.75	0.20
anchor	0.46	0.30	0.36
background_class	0.00	1.00	0.00
barrel	1.00	0.00	0.00
camera	1.00	0.00	0.00

car_side	0.00	0.00	0.00
dalmatian	0.00	0.00	0.00
ferry	1.00	0.00	0.00
headphone	0.19	0.35	0.25
lamp	1.00	0.00	0.00
pizza	0.50	0.05	0.09
pyramid	1.00	0.00	0.00
snoopy	0.33	0.05	0.09
soccer_ball	0.00	1.00	0.00
stop_sign	1.00	0.00	0.00
strawberry	0.12	0.05	0.07
sunflower	0.56	0.25	0.34
water_lilly	0.43	0.15	0.22
windsor_chair	1.00	0.10	0.18
yin_yang	0.67	0.10	0.17
Accuracy			0.14
Macro Average	0.50	0.22	0.11
Weighted Average	0.56	0.14	0.12



K=50

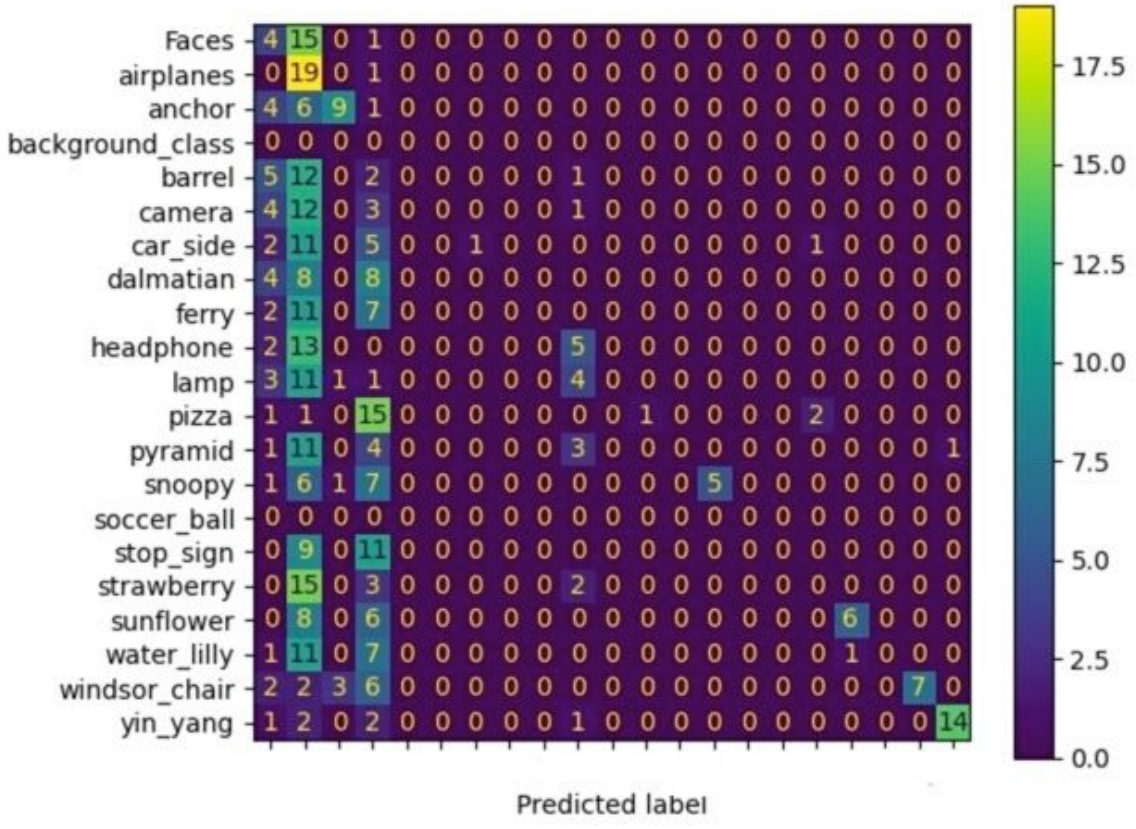
Class	Precision	Recall	F1 Score
Faces	0.15	0.35	0.21
airplanes	0.11	0.85	0.20
anchor	0.45	0.25	0.32
barrel	0.00	1.00	0.00
camera	1.00	0.00	0.00
car_side	1.00	0.00	0.00
dalmatian	0.33	0.05	0.09
ferry	0.50	0.05	0.09
headphone	1.00	0.00	0.00
lamp	0.22	0.30	0.26
pizza	1.00	0.00	0.00
pyramid	1.00	0.00	0.00
snoopy	1.00	0.00	0.00
soccer_ball	1.00	0.25	0.40
stop_sign	0.00	0.00	0.00
strawberry	0.36	0.20	0.26
sunflower	0.50	0.20	0.29
water_lilly	0.00	0.00	0.00
windsor_chair	1.00	0.35	0.52
yin_yang	0.91	0.50	0.65
Accuracy			0.18
Macro Average	0.58	0.22	0.16
Weighted Average	0.61	0.18	0.17



K=100

Class	Precision	Recall	F1 Score
Faces	0.11	0.20	0.14
airplanes	0.10	0.95	0.19
anchor	0.64	0.45	0.53
barrel	0.00	1.00	0.00
camera	1.00	0.00	0.00
car_side	1.00	0.00	0.00
dalmatian	1.00	0.05	0.10
ferry	1.00	0.00	0.00
headphone	1.00	0.00	0.00
lamp	0.29	0.25	0.27
pizza	1.00	0.00	0.00
pyramid	1.00	0.05	0.10
snoopy	1.00	0.00	0.00
soccer_ball	1.00	0.25	0.40
stop_sign	1.00	0.00	0.00

strawberry	0.00	0.00	0.00
sunflower	0.86	0.30	0.44
water_lilly	1.00	0.00	0.00
windsor_chair	1.00	0.35	0.52
yin_yang	0.93	0.70	0.80
Accuracy			0.19
Macro Average	0.75	0.23	0.17
Weighted Average	0.79	0.19	0.18



2.ORB-K-Means(Ipek's Implementation)-Bag of Words(Mahmut's Implementation)-Decision Tree

Author: Mahmut Karaca

By observing the results of classification, we can conclude that decision tree classification is practically useless in our use case. We speculate that a larger dataset might produce better results.

K=20

Class	Precision	Recall	F1 Score
Faces	0.04	0.1	0.06
airplanes	0.07	0.3	0.12
anchor	0.21	0.25	0.23
background_class	0	1	0
barrel	0	0	0
camera	0	0	0
car_side	0.19	0.15	0.17
dalmatian	0	0	0
ferry	0	0	0
headphone	0.07	0.1	0.08
lamp	0	0	0
pizza	0	0	0
pyramid	0.43	0.15	0.22
snoopy	0.25	0.15	0.19
soccer_ball	0	1	0
stop_sign	0.14	0.05	0.07
strawberry	0.24	0.25	0.24
sunflower	0	0	0
water_lilly	0.05	0.05	0.05
windsor_chair	0.25	0.05	0.08
yin_yang	0.2	0.05	0.08

accuracy			0.09
macro_avg	0.1	0.17	0.08
weighted_avg	0.11	0.09	0.08

K=50

Class	Precision	Recall	F1 Score
Faces	0.02	0.05	0.02
airplanes	0.07	0.35	0.12
anchor	0.19	0.2	0.2
background_class	0	1	0
barrel	0	0	0
camera	0	0	0
car_side	0.11	0.05	0.07
dalmatian	0	0	0
ferry	0	0	0
headphone	0.04	0.05	0.04
lamp	0	0	0
pizza	0.2	0.05	0.08
pyramid	0.33	0.05	0.09
snoopy	0.17	0.1	0.12
soccer_ball	0	1	0
stop_sign	0	0	0
strawberry	0	0	0
sunflower	0	0	0
water_lilly	0.06	0.05	0.05
windsor_chair	0.4	0.1	0.16

yin_yang	0.43	0.15	0.22
accuracy			0.06
macro_avg	0.1	0.15	0.06
weighted_avg	0.11	0.06	0.06

K=100

Class	Precision	Recall	F1 Score
Faces	0.11	0.3	0.16
airplanes	0.06	0.25	0.09
anchor	0.12	0.15	0.14
background_class	0	1	0
barrel	0	0	0
camera	0	0	0
car_side	0.22	0.1	0.14
dalmatian	0	0	0
ferry	0.14	0.05	0.07
headphone	0.03	0.05	0.04
lamp	0	0	0
pizza	0	0	0
pyramid	0	0	0
snoopy	0.57	0.2	0.3
soccer_ball	0	1	0
stop_sign	0.29	0.1	0.15
strawberry	0.08	0.05	0.06
sunflower	0	0	0
water_lilly	0.14	0.15	0.15

windsor_chair	0	0	0
yin_yang	0	0	0
accuracy			0.07
macro_avg	0.08	0.16	0.06
weighted_avg	0.09	0.07	0.07

3.HOG(Ahmet Emin's Implementation)-K-Means(Ipek's Implementation)-Bag of Words(Mahmut's Implementation)-SVM

Author: Ahmet Emin Yetkin

K=20
K=50
K=100

4.LBP(Ahmet's Implementation)-K-Means(Ipek's Implementation)-Bag of Words(Mahmut's Implementation)-Adaboost

Author: Ahmet Karagöz

K=20
K=50
K=100

5.Hu Moment(Yasin's Implementation)-K-Means(Ipek's Implementation)-Bag of Words(Mahmut's Implementation)-MLP

Author: Muhammed Yasin Adıyaman

K=20
K=50
K=100

6.Overall Results

Pipeline	K	Macro Average F1 Score
SIFT-K-Means (Ipek) - Bag of Words(Mahmut) - Random Forest	20	0.11
	50	0.16
	100	0.17
ORB-K-Means(Ipek)-Bag of Words(Mahmut)-Decision Tree	20	0.08
	50	0.06
	100	0.06
HOG(Ahmet Emin)-K-Means(Ipek)-Bag of Words(Mahmut)-SVM	20	
	50	
	100	
LBP(Ahmet)-K-Means(Ipek)-Bag of Words(Mahmut)-Adaboost	20	
	50	
	100	
Hu Moment(Yasin)-K-Means(Ipek)-Bag of Words(Mahmut)-MLP	20	
	50	
	100	

References

[1] A. Bosch, A. Zisserman and X. Munoz, "Image Classification using Random Forests and Ferns," *2007 IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, 2007, pp. 1-8, doi: 10.1109/ICCV.2007.4409066.

[2] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi and S. Homayouni, "Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 6308-6325, 2020, doi: 10.1109/JSTARS.2020.3026724.