Due: 5th Apr 2022 11:59 PM Total points: **20**

In this assignment you need to classify images. You will learn a classifier from the training data provided, then use it to classify test images, as described in Section 1. In Section 2 you will be challenged to implement specific improvement(s) to enhance the scalability of your classification algorithm.

Two datasets are provided within `Assignment_08.zip`. Both include training and test data. One is a reduced dataset with 3 classes, while the other is a more expansive 25 class dataset. It is STRONGLY recommended that you begin with the reduced dataset. Once you are satisfied with how your algorithm performs under the reduced class circumstances, then you can explore how your algorithm scales with more classes.

1 Bag of Features Classification with SIFT Descriptors

A *Bag of Features* algorithm uses image features for classification[1]. It operates under the key assumption that the presence or absence of certain features within an image indicate the class of the image. For instance, the presence of "tire" features would indicate membership of the "vehicle" class. SIFT[2] features are in general robust to a wide range of non-ideal imaging situations, and therefore are what you will use for your implementation. This means you will need to find SIFT features and compute their associated feature descriptors for a large pool of images (to help with automatic file handling, see Matlab's `dir(...)`). You can implement the SIFT algorithm yourself, or you are free to use VLFeat's Matlab implementation.

Installation instructions are here:
`http://www.vlfeat.org/install-matlab.html`
And formal documentation of their SIFT function is here:
`http://www.vlfeat.org/overview/sift.html`

SIFT features and descriptors can be found for a single precision gray scale image `I` using `[f,d] = vl_sift(I);`. The output `f` stores the location, size, and orientation of the keypoints while `d` stores the actual (128×1) feature descriptors.

[1]Also sometimes referred to as a *Bag of Words* algorithm, from the techniques by the same name used to classify written documents.

[2]D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004.

1.1 Algorithm Implementation

7.5 points

1. Use SIFT to find features (and their descriptors) in all of the training set images. Do this carefully (see Matlab's `struct` data-type), so that you can easily identify which features belonged to which training class.

2. Cluster all the SIFT feature descriptors you found using a distance metric of your choice (be sure to include your chosen metric in your report along with your rationale). We suggest using the *k-means* clustering algorithm (such as Matlab's `kmeans(...)`), but you are free to use other clustering algorithms if you so desire (be sure to describe your chosen algorithm in your report). About $N = 1000$ clusters is reasonable for the reduced training data provided (3 classes, 50 images each, on the order of hundreds of features per image). Conceptually, you can imagine that each cluster represents the presence of certain distinct "things" in your images. By extension, the more classes you have, the more clusters you should anticipate.

3. (a) For each training class, form a histogram of N bins, where each bin corresponds to a cluster found above. Find how many SIFT features from that class's training images belong to each of the SIFT descriptor clusters (in accordance once again with a distance metric of your choice, see Matlab's `kmeans(...)` or `knnsearch(...)`). In essence, you are associating the "things" you identified during clustering to actual classes, i.e. the presence of tires is characteristic of the car class, the presence of apples is characteristic of the fruit class, etc. Keep in mind that there will be features present which do not help in identifying classes (such as useless junk in the image's background for instance). These "bad" features should be removed with a cluster distance threshold.

   (b) Normalize the bin counts by dividing by the total number of SIFT features binned for that particular class. This normalized histogram now forms a descriptor for that particular training class.

   Now to test your classification algorithm:

4. Find the SIFT feature descriptors within each test image.

5. Assign these features to clusters as you did when creating the class descriptors (removing "bad" features with a cluster distance threshold as before).

6. Use these assignments to calculate a normalized cluster histogram for each test image.

7. Assign each test image to one of the possible classes by comparing its cluster histogram to the cluster histograms of the various classes you trained with previously. One conceptual way to do this comparison is to think of each normalized cluster histogram as a vector. Finding the class is akin to finding the class histogram vector closest (with respect to your chosen distance metric) to the one associated with image being tested. This can be implemented using Matlab's `knnsearch`.

1.2 Technical Write-up: Results and Discussion

7.5 points

- Clearly and cogently document your methods and results. From your PDF report, it should be clear what you did, how/why you did it, and how well it worked, without needing to run code or sift through 300 figures.
- Indicate points of possible improvement and provide conceptual solutions to the extent you are able.
- Include and interpret a (3×3) confusion matrix (for your results on the reduced dataset). See below for an example:

		Predicted class		
	Classes	Hat	Butterfly	Airplane
Actual class	Hat	55%	31%	14%
	Butterfly	20%	70%	10%
	Airplane	12%	5%	83%

1.3 Competition

5 points

The average value of the diagonal elements in the confusion matrix above is the accuracy of the classifier. For the above case, the accuracy is 69.33%. Report the accuracy of your classifier. This part of the grade is based on your classifier's performance compared to the classifiers trained by your peers in the class.
Note: Failure to report this number will automatically award you zero points and reporting wrong numbers is against the honor code.

Submission Instructions

Every student must submit following 2 files:

- An organized report submitted as a PDF document. The report should describe the implementation, issues (problems encountered, surprises), and an analysis of the test results (interpretation of effects of varying parameters, different image results). Intermediate and final results must be provided.
- A ZIP file containing the necessary codes.

The heading of the PDF file should contain the assignment number and topic. Also, attach a photo of yourself at top-left of the PDF along with your name and department.

Late Submission Policy

Assignments are expected to be submitted on the due date. Each student gets a total of 3 late days that can be used however you wish. For examples, all 3 days can be used towards 1 assignment or 1 day late for 3 assignments or other combinations. Late submissions beyond that will be penalized as below:

- One day late will be penalized 25% of the credit.
- Two Days late will be penalized 50%.
- Submissions more than 2 days late will not be considered for credit.

I will be ruthless in enforcing this policy. There will be no exceptions

Collaboration Policy

I encourage collaboration both inside and outside class. You may talk to other students for general ideas and concepts but the programming must be done independently. For mid-term and final examination there will be no collaboration permitted.

Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly. If you have any doubts regarding what is and is not plagiarism, talk to me.