ECE 479/5582 Computer Vision Homework -2 Gradient Features and Aggregation

Student Name: Christopher Seagraves Student ID:____________________

Data set: the NWPU aerial data set contains approximately 45 categories of 700 images for each category in 256x256 RGB format. The data set can be downloaded from:

https://umkc.box.com/s/fxvzh5qq2tiob6eklfxfwn89kg3e1io1

For HW-2, let us just take the first 15 classes (shown above) {airplane, airport, bridge, ..., freeway} from the full data set, and 60 images per class and form a data set of N=900 images with 15 labels. Objectives are using the gradient image features (SIFT, Dense SIFT) with aggregation (VLAD, FisherVector) to test which one can give us the best solution.

[1] Gradient Features, for each image, implement the following function in either Matlab and Python, show your implementation [25pts]

```
% im - input images, let us make them all grayscale only, so it is a h x w matrix
% opt.type = { 'sift' , 'hogf' , 'dsft' } for sift and densesift
% f - n x d matrix containing n features of d dimension, d=128 for SIFT, for e.g,
function [f]=getImageFeatures(im, opt)
......
```

Name: _________________ Id: _________________ Email:___________________

Student Name: Christopher Seagraves Student ID:____________________

```
if COMPUTE_FEATURES

    for i = 1:numel(labels)
        label = labels{i};
        label_images = images(label);
        fprintf('Processing %s images...\n', label);

        for j = 1:numel(images(label))
            image = label_images(j);

            [x, sift_features] = get_image_features(single(image.grayscale_image), ImageFeatureOperations.SIFT);
            f_sift = [f_sift, sift_features];
            image.sift_features = sift_features;

            [x, dsift_features] = get_image_features(single(image.grayscale_image), ImageFeatureOperations.DSIFT);
            f_dsift = [f_dsift, dsift_features];
            image.dsift_features = dsift_features;

            label_images(j) = image;
        end

        images(label) = label_images;

    end

    fprintf('Saving features...\n')
    save('images.mat', 'images', '-v7.3')
```

```
function [keypoints, descriptors] = get_image_features(image, operation)

    if strcmp(operation, ImageFeatureOperations.SIFT)
        [keypoints, descriptors] = vl_sift(image);
    elseif strcmp(operation, ImageFeatureOperations.DSIFT)
        [keypoints, descriptors] = vl_dsift(image);
    else
        error('Invalid operation');
    end

end
```

Save features as a cell structure, f_sift{}, f_dsift{}, for N images, submit as a gradient_features.mat file. Use cell structure.

Early in the homework I did save a gradient_features.mat file, but after converting some of my HW1 stuff from Python to MATLAB, I ended up making an Image class and just put the features in there and saved to images.mat throughout the process.

Name: _________________ Id: _________________ Email:___________________

Student Name: Christopher Seagraves Student ID:___________________

```
classdef Image

    properties
        path
        label
        rgb_image
        name
        hsv_image
        flattened_hsv_image
        grayscale_image
        sift_features
        hog_features
        dsift_features
        sift_fisher_vectors
        dsift_fisher_vectors
    end

    methods

        function obj = Image(path, label, rgb_image)
            obj.path = path;
            obj.label = label;
            obj.rgb_image = rgb_image;

            obj.name = strsplit(path, '/');
            obj.name = obj.name{end};
            obj.hsv_image = rgb2hsv(rgb_image);
            obj.flattened_hsv_image = reshape(obj.hsv_image, [], 3);
            obj.grayscale_image = rgb2gray(rgb_image);

            obj.sift_features = [];
            obj.hog_features = [];
            obj.dsift_features = [];

            obj.sift_fisher_vectors = {};
            obj.dsift_fisher_vectors = {};
        end

    end

end
```

Student Name: Christopher Seagraves Student ID:____________________

[2] Compute the PCA and GMM model for each feature space:[25 pts]

(2a, 10pts), implement the following function that computes various gradient features:

```
% f - n x d matrix containing n features from say 100 images.
% nc - number of GMM components
% kd - desired lower dimension of the feature
% fv_gmm - FisherVector GMM model:
%       fv_gmm.m - mean, fv_gmm.cov - variance, fv_gmm.p - prior
% A - dxd PCA for dimension reduction
function [gmm, A]=getFisherVectorModel(f, kd, nc)
[A,s,lat]=princomp(f);
f0 = f*A(:,1:kd); % this is the feature with desired d-dimensions
% call vl_gmm here: gmm.mean, gmm.var, gmm.prior are kd x nc dimensions
.....
return;
```

Student Name: Christopher Seagraves Student ID:____________________

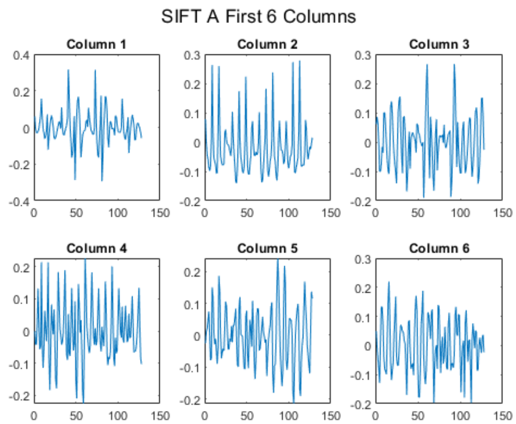
```
function [A, sift_gmm] = train_sift_gmm(sift_train_gmm, kds, ncs, operation)
    dbg = 0;

    if dbg
        data = 'data';
        addpath(genpath([pwd, filesep, data]));
        load([data, filesep, 'sift_train_gmm.mat']);
        kds = [8 12 16 24 32];
        ncs = [16, 24, 32, 64, 128];
    end

    tic;
    [A, s, lat] = pca(double(sift_train_gmm));
    t1 = toc;

    for j = 1:length(kds)
        x = double(sift_train_gmm) * A(:, 1:kds(j)); % data projected

        for k = 1:length(ncs)
            fprintf('\n t=%1.2f training gmm(%d %d)', toc, kds(j), ncs(k));
            [sift_gmm(j, k).mean, sift_gmm(j, k).cov, sift_gmm(j, k).prior, sift_gmm(j, k).ll] = vl_gmm(x', ncs(k), 'MaxNumIterations', 200);
            fprintf(' ll = %1.2f ', sift_gmm(j, k).ll);
        end % for k

    end % for j

    if strcmp(operation, ImageFeatureOperations.SIFT)
        fprintf('Saving SIFT GMM...\n')
        sift_A = A;
        save('sift_gmm.mat', 'sift_A', 'sift_gmm', '-v7.3');
    elseif strcmp(operation, ImageFeatureOperations.DSIFT)
        fprintf('Saving dense SIFT GMM...\n')
        dsift_gmm = sift_gmm;
        dsift_A = A;
        save('dsift_gmm.mat', 'dsift_A', 'dsift_gmm', '-v7.3');
    else
        error('Invalid operation');
    end

    % if dbg
    %     figure(41); title('SIFT GMM (8, 16)'); grid on; hold on;
    %     plot3d(sift_gmm(1, 1).mean', '*r');
    % end

    return;
end
```

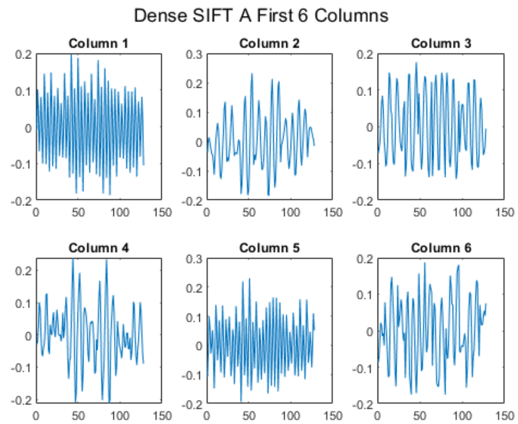
```
    if TRAIN_GMMS
        [sift_A, sift_gmm] = train_sift_gmm(f_sift_flattened, kds, ncs, ImageFeatureOperations.SIFT);
        [dsift_A, dsift_gmm] = train_sift_gmm(f_dsift_flattened, kds, ncs, ImageFeatureOperations.DSIFT);
    else
```

(2b, 5pts): Visualize the eigen values for SIFT, and Dense SIFT features here, example of SIFT is shown in lecture notes.

Name: ________________ Id: ________________ Email:___________________

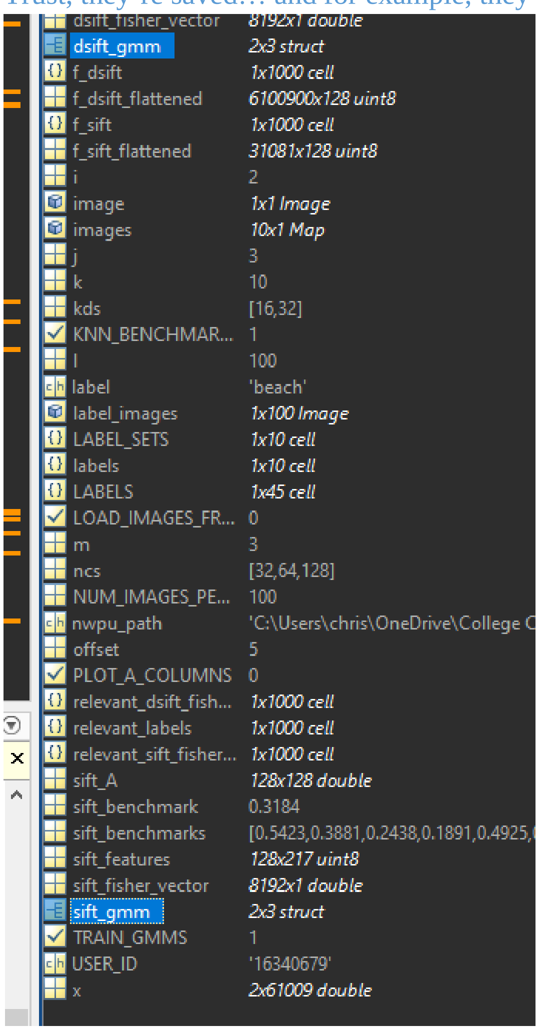
Student Name: Christopher Seagraves Student ID:____________________





(2c, 10pts): Save the GMM models for SIFT, DenseSIFT features for a combination of PCA dimensions kd=[16, 32], and GMM components nc=[32, 64, 128]. {Hint: example code and training data provided}

Trust, they're saved… and for example, they're a 2x3 struct to correlate with the kds and ncs



Student Name: Christopher Seagraves Student ID:___________________

[3] Compute the gradient feature aggregations and benchmark its recognition accuracy. [50pts]

(3a: 20pts), implement the FisherVector aggregation function here based on vl_feat library:

function [fv]=**getFisherVector**(f, A, gmm, kd, nc)

% f: n x d features

%A: d x d PCA matrix

% gmm: gmm.mean, gmm.var, gmm.prior, the GMM of dimension kd by nc

……

```
if COMPUTE_FISHER_VECTORS

    for i = 1:numel(labels)
        label = labels{i};
        label_images = images(label);
        fprintf('Getting fisher vectors %s images...\n', label);

        for j = 1:numel(images(label))
            image = label_images(j);

            image.sift_fisher_vectors = {};
            image.dsift_fisher_vectors = {};

            for k = 1:size(sift_gmm, 1)

                for m = 1:size(sift_gmm, 2)
                    sift_fisher_vector = get_fisher_vector(image.sift_features', sift_A, sift_gmm(k, m), kds(k), ncs(m));
                    dsift_fisher_vector = get_fisher_vector(image.dsift_features', dsift_A, dsift_gmm(k, m), kds(k), ncs(m));
                    image.sift_fisher_vectors = [image.sift_fisher_vectors, sift_fisher_vector];
                    image.dsift_fisher_vectors = [image.dsift_fisher_vectors, dsift_fisher_vector];
                end

            end


            label_images(j) = image;

        end

        images(label) = label_images;

    end

    fprintf('Saving fisher vectors...\n')
    save('images.mat', 'images', '-v7.3')
end
```

```
function [fisher_vector] = get_fisher_vector(features, A, gmm, kd, nc)
    f0 = double(features) * A(:, 1:kd);
    fisher_vector = vl_fisher(f0', gmm.mean, gmm.cov, gmm.prior);
end
```

(3b: 20 pts), compute the knnclassify() based recognition (leave 1 out) results and confusion map, just like in HW-1 here for a combination of features, GMM sizes, fill in the table below, and also show confusion map:

Name: _______________ Id: _______________ Email:_________________

Student Name: Christopher Seagraves Student ID:____________________

SIFT FV aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

GMM Size (Kd/nc)	32	64	128
16	54%	39%	24%
32	19%	49%	32%

Dense SIFT FV aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

GMM Size (Kd/nc)	32	64	128
16	66%	62%	64%
32	66%	69%	67%

HoG FV aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

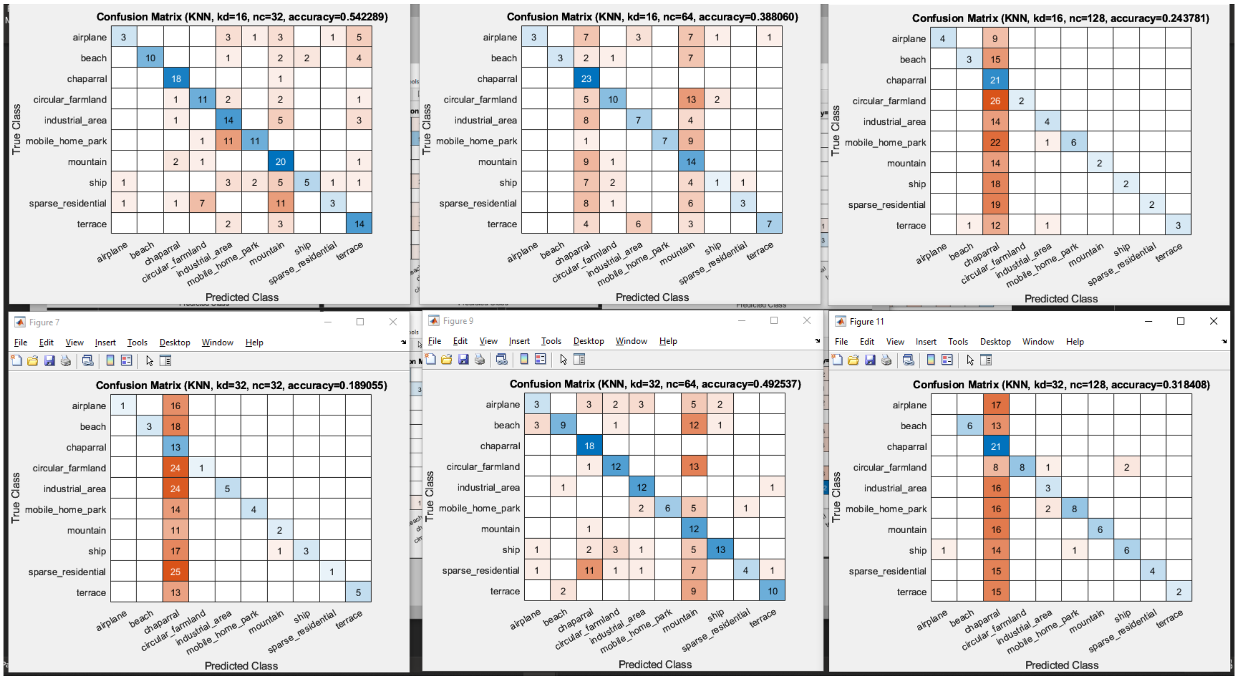
GMM Size (Kd/nc)	32	64	128
16			
32			

Name: _________________ Id: _________________ Email:___________________

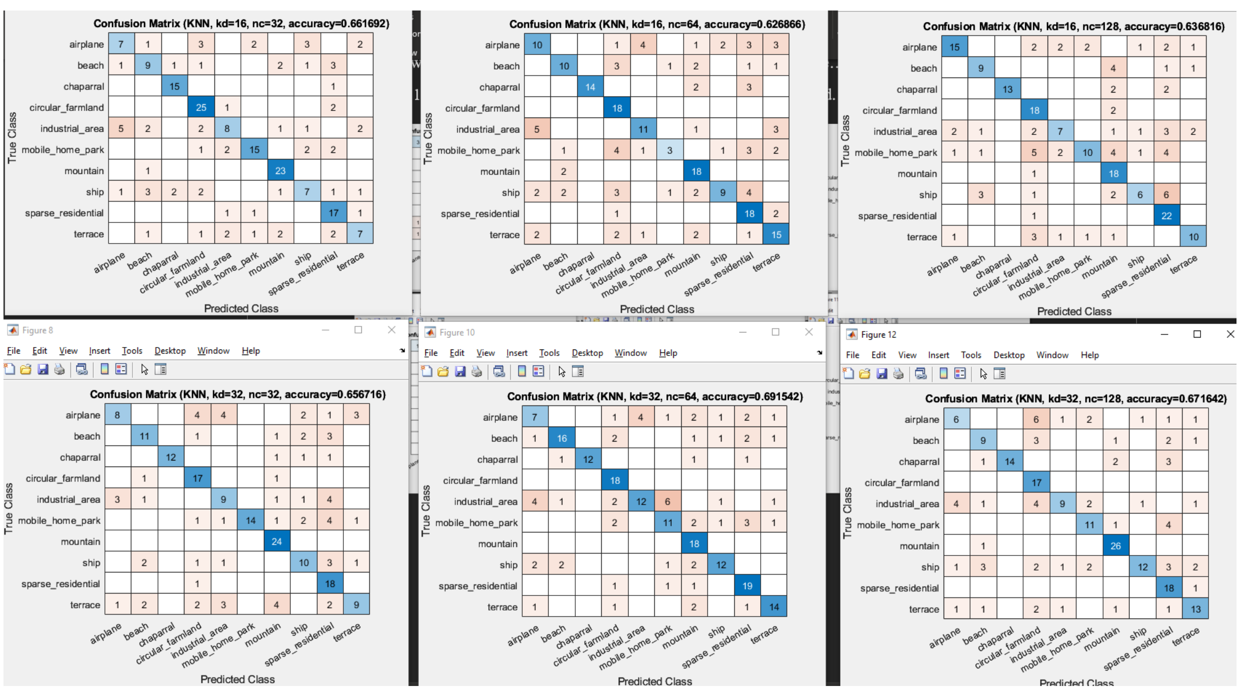
Student Name: Christopher Seagraves Student ID:___________________

(3c: 10 pts), Plot the confusion map for 6 combinations for each feature, one example for SIFT, kd=16, nc=64 is shown below: {Hint: confusionmat()}

Not sure what happened with predicting chaparral overload on 3 of these… This was 100 images per label, and using 80% train-test split, but yeah, odd.



For comparison, here's the dense sift



Student Name: Christopher Seagraves Student ID:____________________

[4] Extra Credit: Compute the gradient feature VLAD aggregations and benchmark its recognition accuracy. [30pts]

(4a: 15pts), implement the VLAD aggregation function here based on vl_feat library:

```
function [fv]=getVLAD(f, gmm, kd, nc)

% use the GMM means as kmeans centroids

......
```

(4b: 15pts), compute the knnclassify() based recognition (leave 1 out) results and confusion map, just like in HW-1 here for a combination of features, GMM sizes, fill in the table below, and also show confusion map:

SIFT VLAD aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

GMM Size (Kd/nc)	32	64	128
16			
32			

Dense SIFT VLAD aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

GMM Size (Kd/nc)	32	64	128
16			
32			

HoG VLAD aggregation recognition accuracy: sum(diag(cm))/sum(cm(:))

GMM Size (Kd/nc)	32	64	128
16			
32			

Student Name: Christopher Seagraves Student ID:____________________