Name: Neil Chen December 1, 2021

PID: A59013393

ECE 271A Homework 5

1. This week we use the cheetah image to evaluate the performance of a classifier based on mixture models estimated with EM. Once again, we use the decomposition into 8 x 8 image blocks, compute the DCT of each block, and zig-zag scan. For this (using the data in **TrainingSamplesDCT new 8.mat**) we fit a mixture of Gaussians of diagonal covariance to each class, i.e. $P_{X|Y}(x|i) = \sum_{c=1}^{C} \pi_c G(x, \mu_c, \sigma_c)$ where all σ_c are diagonal matrices. We then apply the BDR based on these density estimates to the cheetah image and measure the probability of error as a function of the number of dimensions of the space (as before, use $\{1, 2, 4, 8, 16, 24, 32, \dots, 64\}$ dimensions).

   (a) For each class, learn 5 mixtures of C = 8 components, using a random initialization (recall that the mixture weights must add up to one). Plot the probability of error vs. dimension for each of the 25 classifiers obtained with all possible mixture pairs. Comment the dependence of the probability of error on the initialization.

Solution.

In order to perform EM for Gaussian mixtures, we need three variables, that are: π, μ, and $\sum$. The first step is to initialize all the variables. Firstly, we create the parameter π. Creating it is easy, we just create a matrix of size C, each entry is less than 1, and the sum of the matrix equals to 1. Secondly, we create the parameter μ. We randomly select C observations from the training data set. Lastly, we create a positive definite matrix for the parameter $\sum$. For simplicity, we create a random vector and diagonalize it and make it our $\sum$. After setting up the parameters, we now move on to the E step of our learning. We calculate the h_{ij} variable by using the formula mentioned below.

$$h_{ij} = \frac{G(x_i, u_j^{(n)}, \sigma_j^{(n)})\pi_j^{(n)}}{\sum_{k=1}^{C} G(x_i, u_j^{(n)}, \sigma_j^{(n)})\pi_j^{(n)}} \tag{1}$$

With the h_{ij} value known, we can proceed to the M step. $\psi^{(n+)}$ can be calculated by using the formula mentioned below.

$$\psi^{(n+)} = \arg\min_{\psi} \sum_{ij} h_{ij} \log[G(x_i, u_j^{(n)}, \sigma_j^{(n)})] \tag{2}$$

We want to optimize the formula by using Lagrangian (per the note in the lecture). The formula we need for updating our parameters are listed below.

$$u_j^{(n+1)} = \frac{\sum_i h_{ij} x_i}{\sum_i h_{ij}} \tag{3}$$

$$\pi_j^{(n+1)} = \frac{1}{n} \sum_i h_{ij} \tag{4}$$

$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij}(x_i - u_j)^2}{\sum_i h_{ij}} \tag{5}$$

With those formulae in hand, we have no problem updating our parameters. We set a maximum number of time for update, **EMLimit** = 1000. If the number of iteration exceeds it, then our algorithm will break. Also, we compare our result from the current iteration with that of the last iteration. If the value of the likelihood does not change by 0.001, then we will break our algorithm as well. Since there is no sense running the algorithm if the value does not differ. One crucial thing to keep in mind is that $\sum$ should always be positive definite; otherwise, the algorithm will crash. To achieve this, we add a small value to $\sum$ after each iteration. By doing this, we can assure that $\sum$ remains positive definite at all time.

Now **Part a** asks us to train 5 EMs for *cheetah* and 5 EMs for *grass* and then plot the error rates and compare them, i.e., perform a cross-comparison among those 25 EMs. It is an arduous task but after hours of running the algorithm, here are my results. I plot the first EM of *cheetah* with five EMs of *grass*, then plot the second EM of *cheetah* with five EMs of *grass*, etc. From **Figure 1 (a)** to **Figure 1 (k)**, we can confidently say that as the value of dimensions increase, so does the performance. We can be more precise by observing the plot from **Figure 2** to **Figure 6**. The error rates decrease dramatically. There exhibit similar results across all those 25 EMs (5 EMs for *cheetah* and 5 for *grass*).

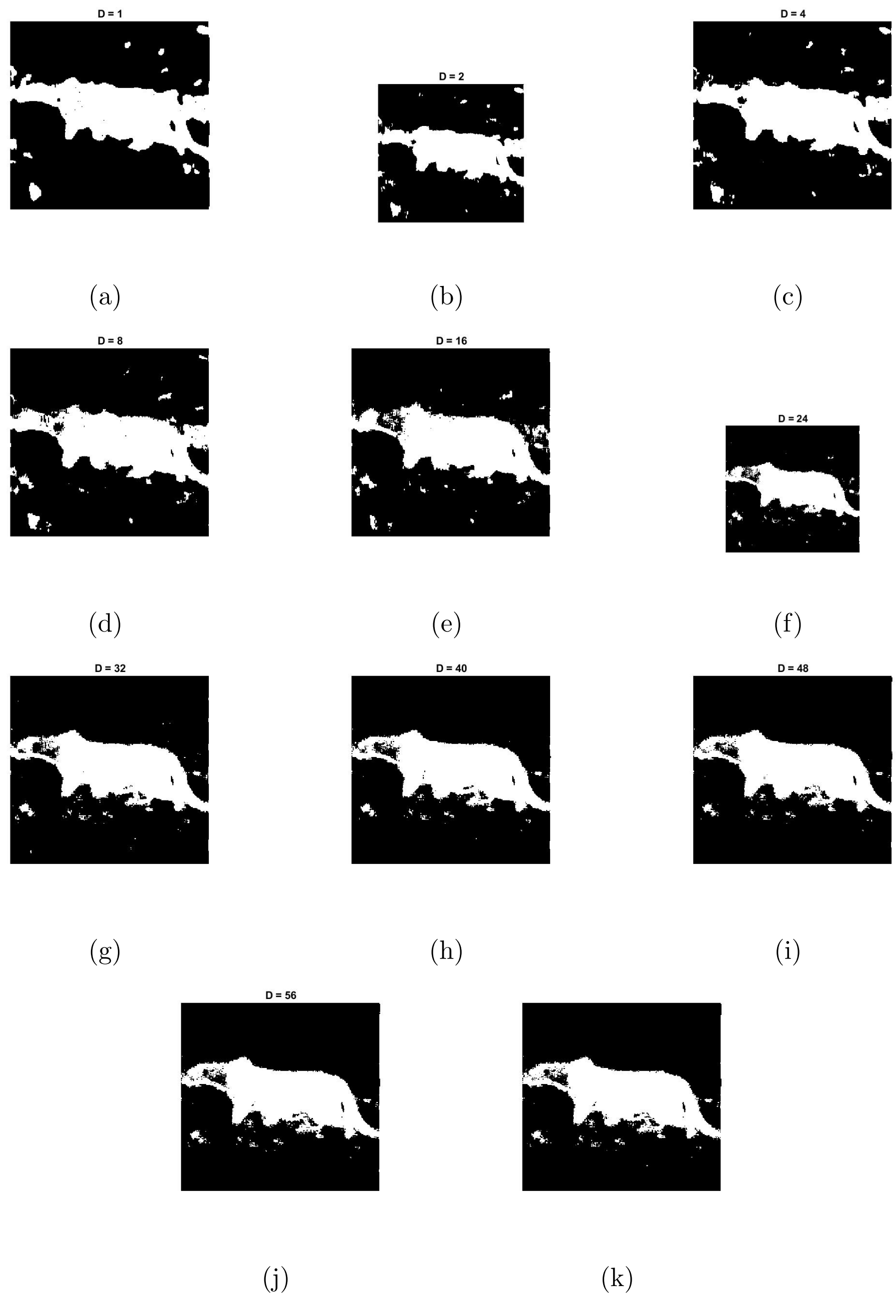

Figure 1: Results from EM: (a) D = 1 (b) D = 2 (c) D = 4 (d) D = 8 (e) D = 16 (f) D = 24 (g) D = 32 (h) D = 40 (i) D = 48 (j) D = 56 (k) D = 64

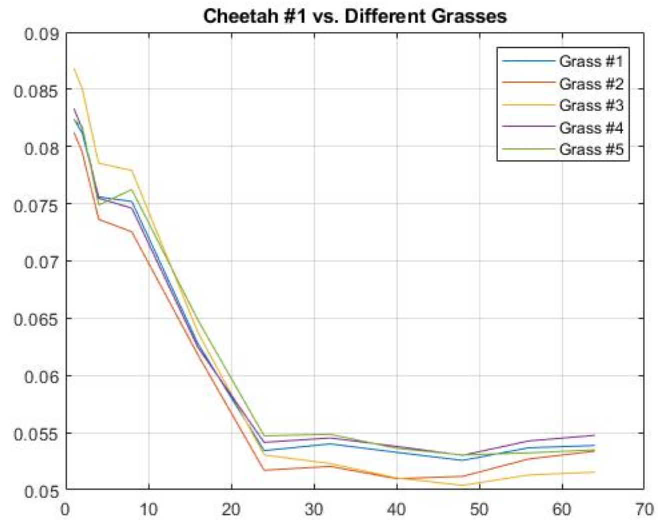

Figure 2: Cheetah #1 vs. Grasses

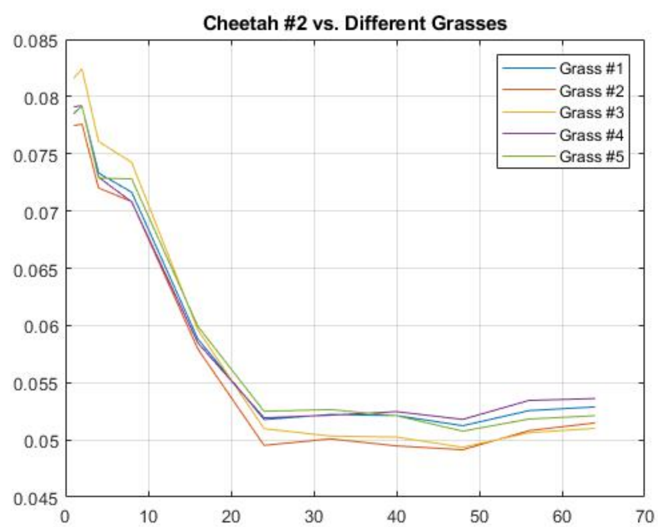

Figure 3: Cheetah #2 vs. Grasses

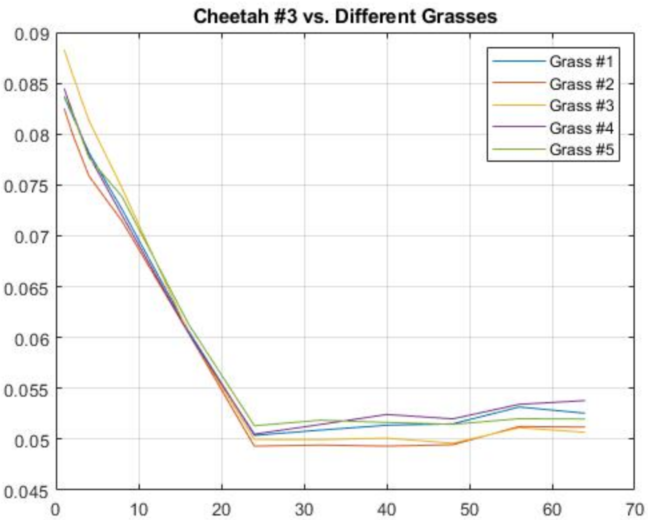

Figure 4: Cheetah #3 vs. Grasses

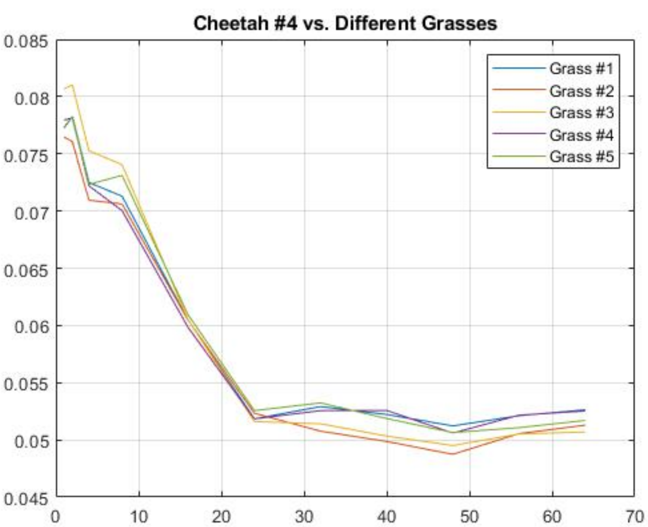

Figure 5: Cheetah #4 vs. Grasses

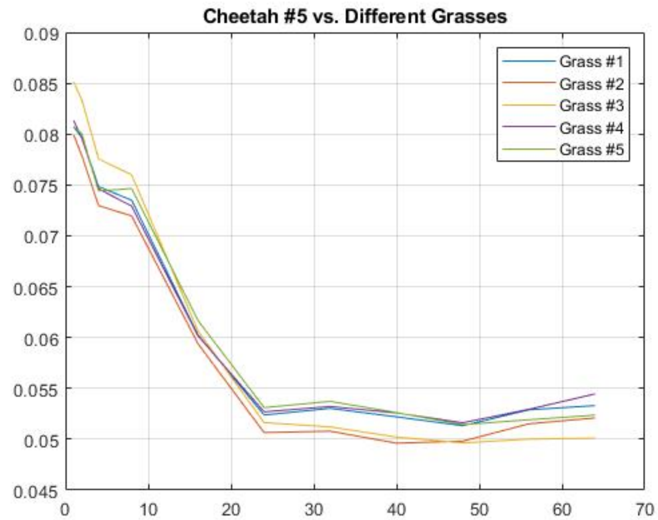

Figure 6: Cheetah #5 vs. Grasses

(b) For each class, learn mixtures with C $\in$ {1, 2, 4, 8, 16, 32}. Plot the probability of error vs. dimension for each number of mixture components. What is the effect of the number of mixture components on the probability of error?

Solution.
After learning mixture models for $C = 1, 2, 4, 8, 16, 32$, we plot the result as shown in **Figure 7**. We can tell that when $C = 1$ and $C = 2$, the error rates are high. Nevertheless, when we look at C that are greater than or equal to 4, the EM algorithm yields decent results. And the results from all of them are quiet similar when the value of the dimensions is low. They only perform slightly different when the value of the dimensions is high, i.e., when d is greater than or equal to 32.

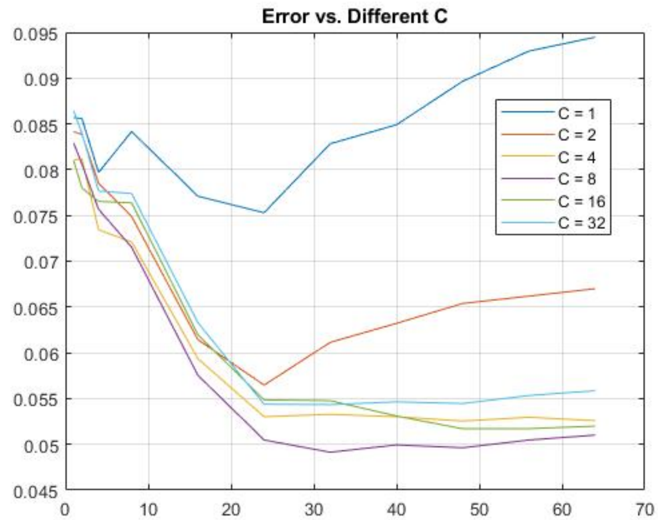

Figure 7: Error Rate vs. Different Dimensions

Code for EM (Mother Code)

```
clear
clc

% load all the data
load("ZigZagVec.mat")    % 1 x 64 vector
load("cheetahMat.txt")  % our image
load("TrainingSamplesDCT_8_new.mat")     % the training data
    sets
load("zigzagImg.mat") % the preprocessed image
load("cheetah_mask.mat")     % the preprocessed mask

% dimensions for the training & the parameters for EM
C = 8; % number of components
dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of
    dimensions of the space

% # of samples
[n_cheetah, ~] = size(TrainsampleDCT_FG);
[n_grass, ~]   = size(TrainsampleDCT_BG);
totalSamples = (n_cheetah + n_grass);

% get the size of the image
```

```matlab
[m, n] = size(cheetahMat);

% create a matrix
m = m - 7;
n = n - 7;
totalPixels = m * n;

% ---- EM for cheetah ----

% initialize pi by creating a random normalized 1 x C matrix
pi_cheetah = randi(1, C);
pi_grass   = randi(1, C);
pi_cheetah = pi_cheetah / sum(pi_cheetah);
pi_grass   = pi_grass   / sum(pi_grass);

% initialize mu by choosing C samples randomly from the training data set
p = randperm(n_cheetah, C);     % pick C random numbers from 0 to the size of the sample of cheetah
mu_cheetah = zeros(C, 64);  % create an array for mu
mu_grass   = zeros(C, 64);  % create an array for mu
for i = 1 : C
    mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
    mu_grass(i, :)   = TrainsampleDCT_BG(p(i), 64);
end

% initialize sigma by creating a diagonal matrix with random variable
sigma_cheetah = zeros(64, 64, C);
sigma_grass   = zeros(64, 64, C);
for i = 1 : C
    randomVec = rand(1, 64);     % generate a random vector (1 x 64)
    sigma_cheetah(:, :, i) = diag(randomVec);   % diagonlize the random vector
    randomVec = rand(1, 64);     % generate a random vector (1 x 64)
    sigma_grass(:, :, i) = diag(randomVec);   % diagonlize the random vector
```

```
end

EMLimit = 1000; % maximum iterations

% —— Start of EM for cheetah ——
jointPro_cheetah = zeros(n_cheetah, C);
for i = 1 : EMLimit
    % E step
    for j = 1 : C
        % P_{X, Z}(x, z; psi) = P_{X|Z}(x|z; psi) * P_Z(z; psi)
        jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG, mu_cheetah(j, :), sigma_cheetah(:, :, j)) * pi_cheetah(j); % mvnpdf returns the pdf of given X, mu, sigma
    end

    % calculate the log likelihood of the
    sumRow = sum(jointPro_cheetah, 2);    % returns the sum of the elements in each row
    likelihood_cheetah = sum(log(sumRow));

    % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j, sigma_j) * pi_j)
    hij_cheetah = jointPro_cheetah ./ sumRow;

    % M step

    % update pi for (n+1)
    % pi = 1 / n * (sum(hij))
    pi_cheetah = sum(hij_cheetah) / n_cheetah;

    % update mu for (n+1)
    % mu = sum(hij * x)/sum(hij)
    % TODO: mu_cheetah changes dimension
    mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(hij_cheetah)';

    % update sigma for (n+1)
```

```
    % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
    for j = 1:C
        sigma_cheetah(:,:,j) = diag(diag(((TrainsampleDCT_FG - mu_cheetah(j,:))'.*hij_cheetah(:,j)'* ...
            (TrainsampleDCT_FG - mu_cheetah(j,:)) ./ sum(hij_cheetah(:,j),1)) + 0.0000001));
    end

    % break condition
    % breaks the loop if the likelihood does not change more than 0.001
    if i > 1
        if abs(likelihood_cheetah - likelihood_cheetah_previous) < 0.001
            break;
        end
    end

    % store the current result as the previous
    likelihood_cheetah_previous = likelihood_cheetah;

end
% --- End of EM for cheetah ---

% --- Start of EM for grass ---
jointPro = zeros(n_grass, C);
for i = 1 : EMLimit
    % E step
    for j = 1 : C
        % P_{X, Z}(x, z; psi) = P_{X|Z}(x|z; psi) * P_Z(z; psi)
        jointPro(:, j) = mvnpdf(TrainsampleDCT_BG, mu_grass(j, :), sigma_grass(:, :, j)) * pi_grass(j); % mvnpdf returns the pdf of given X, mu, sigma
    end

    % calculate the log likelihood of the
    sumRow = sum(jointPro, 2);    % returns the sum of the elements in each row
```

```
    likelihood_grass = sum(log(sumRow));

    % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j,
        sigma_j) * pi_j)
    hij_grass = jointPro ./ sumRow;

    % M step

    % update pi for (n+1)
    % pi = 1 / n * (sum(hij))
    pi_grass = sum(hij_grass) / n_grass;

    % update mu for (n+1)
    % mu = sum(hij * x)/sum(hij)
    mu_grass = hij_grass' * TrainsampleDCT_BG ./ sum(hij_grass
        )';

    % update sigma for (n+1)
    % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
    for j = 1:C
        sigma_grass(:, :, j) = diag(diag(((TrainsampleDCT_BG -
            mu_grass(j,:))'.*hij_grass(:,j)'* ...
            (TrainsampleDCT_BG - mu_grass(j,:)) ./ sum(
                hij_grass(:,j),1)) + 0.0000001));
    end

    % break condition
    % breaks the loop if the likelihood does not change more
        than 0.001
    if i > 1
        if abs(likelihood_grass - likelihood_grass_previous) <
            0.001
            break;
        end
    end

    % store the current result as the previous
    likelihood_grass_previous = likelihood_grass;

```

```
end
% —— End of EM for grass ——

% —— BDR ——
lenList = length(dimensionList);
errorMat = zeros(1, lenList);
for curDim = 1 : lenList

    Kth = dimensionList(curDim);

    % compare BDR for EM
    maskVec = zeros(m * n, 1);    % a vector of our mask, will
        resize it later

    for x = 1 : length(zigzagImg)

        % set the probability of each class as zero
        pro_cheetah = 0;
        pro_grass   = 0;

        % compute total BDR for cheetah
        for y = 1:size(mu_cheetah,1)
            pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x, 1
                : Kth), mu_cheetah(y, 1 : Kth), sigma_cheetah(1
                : Kth, 1 : Kth,y)) * pi_cheetah(y);
        end

        % compute total BDR for grass
        for y = 1:size(mu_grass,1)
            pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1 :
                Kth), mu_grass(y, 1 : Kth),sigma_grass(1 : Kth,
                1 : Kth,y))*pi_grass(y);
        end

        % decide whether the pixel is cheetah or grass
        if pro_cheetah > pro_grass
            maskVec(x) = 1;
        end
    end
```

```
    % resize the vector to matrix
    maskMat = Vec2Mat(maskVec);

    % compute the error rate
    errorMat(curDim) = Err(cheetah_mask, maskMat);
    errorRate = errorMat ./ (m * n);
    imshow(maskMat)
    figure
end
```

Code for Part A

```
clear
clc

% load all the data
load("ZigZagVec.mat")    % 1 x 64 vector
load("cheetahMat.txt")  % our image
load("TrainingSamplesDCT_8_new.mat")     % the training data
    sets
load("zigzagImg.mat") % the preprocessed image
load("cheetah_mask.mat")     % the preprocessed mask

% dimensions for the training & the parameters for EM
C = 8; % number of components
dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of
    dimensions of the space

% # of samples
[n_cheetah, ~] = size(TrainsampleDCT_FG);
[n_grass, ~]   = size(TrainsampleDCT_BG);
totalSamples = (n_cheetah + n_grass);

% get the size of the image
[m, n] = size(cheetahMat);

% create a matrix
m = m - 7;
```

```
n = n - 7;
totalPixels = m * n;

% —— EM for cheetah ——

% —— End of EM for cheetah ——

% —— Start of EM for grass ——

% —— End of EM for grass ——

pi_cheetah = zeros(1, 8, 5);
pi_grass   = zeros(1, 8, 5);
mu_cheetah = zeros(8, 64, 5);
mu_grass = zeros(8, 64, 5);
sigma_cheetah = zeros(64, 64, 8, 5);
sigma_grass = zeros(64, 64, 8, 5);

for i = 1 : 5
    [pi_cheetah(:, :, i), mu_cheetah(:, :, i), sigma_cheetah
        (:, :, :, i)] = GenerateCheetah(n_cheetah,
        TrainsampleDCT_FG);
    [pi_grass(:, :, i), mu_grass(:, :, i), sigma_grass(:, :,
        :, i)] = GenerateGrass(n_grass, TrainsampleDCT_BG);
end

% —— BDR ——
errorRate1 = zeros(1, 11, 5);
for i = 1 : 5
    errorRate1(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
        mu_cheetah(:, :, 1), sigma_cheetah(:, :, :, 1),
        pi_cheetah(:, :, 1), mu_grass(:, :, i), sigma_grass(:,
        :, :, i), pi_grass(:, :, i));
end
errorRate2 = zeros(1, 11, 5);
for i = 1 : 5
    errorRate2(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask,
        mu_cheetah(:, :, 2), sigma_cheetah(:, :, :, 2),
        pi_cheetah(:, :, 2), mu_grass(:, :, i), sigma_grass(:,
```

```
        :, :, i), pi_grass(:, :, i));
end
errorRate3 = zeros(1, 11, 5);
for i = 1 : 5
    errorRate3(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask, mu_cheetah(:, :, 3), sigma_cheetah(:, :, :, 3), pi_cheetah(:, :, 3), mu_grass(:, :, i), sigma_grass(:, :, :, i), pi_grass(:, :, i));
end
errorRate4 = zeros(1, 11, 5);
for i = 1 : 5
    errorRate4(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask, mu_cheetah(:, :, 4), sigma_cheetah(:, :, :, 4), pi_cheetah(:, :, 4), mu_grass(:, :, i), sigma_grass(:, :, :, i), pi_grass(:, :, i));
end
errorRate5 = zeros(1, 11, 5);
for i = 1 : 5
    errorRate5(:, :, i) = ComputeBDR(zigzagImg, cheetah_mask, mu_cheetah(:, :, 5), sigma_cheetah(:, :, :, 5), pi_cheetah(:, :, 5), mu_grass(:, :, i), sigma_grass(:, :, :, i), pi_grass(:, :, i));
end


% --- functions ---

% generate a class for cheetah
function [pi_cheetah, mu_cheetah, sigma_cheetah] = GenerateCheetah(n_cheetah, TrainsampleDCT_FG)
    C = 8;
    pi_cheetah = randi(1, C);
    pi_cheetah = pi_cheetah / sum(pi_cheetah);
    p = randperm(n_cheetah, C);
    mu_cheetah = zeros(C, 64);
    for i = 1 : C
        mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
    end
    sigma_cheetah = zeros(64, 64, C);
```

```
    for i = 1 : C
        randomVec = rand(1, 64);
        sigma_cheetah(:, :, i) = diag(randomVec);
    end
    EMLimit = 1000;
    jointPro_cheetah = zeros(n_cheetah, C);
    for i = 1 : EMLimit
        for j = 1 : C
            jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG, mu_cheetah(j, :), sigma_cheetah(:, :, j)) * pi_cheetah(j); % mvnpdf returns the pdf of given X, mu, sigma
        end
        sumRow = sum(jointPro_cheetah, 2);
        likelihood_cheetah = sum(log(sumRow));
        hij_cheetah = jointPro_cheetah ./ sumRow;
        pi_cheetah = sum(hij_cheetah) / n_cheetah;
        mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(hij_cheetah)';
        for j = 1:C
            sigma_cheetah(:,:,j) = diag(diag(((TrainsampleDCT_FG - mu_cheetah(j,:))'.*hij_cheetah(:,j)'* ...
                (TrainsampleDCT_FG - mu_cheetah(j,:)) ./ sum(hij_cheetah(:,j),1)) + 0.0000001));
        end
        if i > 1
            if abs(likelihood_cheetah - likelihood_cheetah_previous) < 0.001
                break;
            end
        end
        likelihood_cheetah_previous = likelihood_cheetah;
    end
end

% generate a class for grass
function [pi_grass, mu_grass, sigma_grass] = GenerateGrass(n_grass, TrainsampleDCT_BG)
```

```
    C = 8;
    pi_grass = randi(1, C);
    pi_grass = pi_grass / sum(pi_grass);
    p = randperm(n_grass, C);
    mu_grass = zeros(C, 64);
    for i = 1 : C
        mu_grass(i, :) = TrainsampleDCT_BG(p(i), 64);
    end
    sigma_grass = zeros(64, 64, C);
    for i = 1 : C
        randomVec = rand(1, 64);
        sigma_grass(:, :, i) = diag(randomVec);
    end
    EMLimit = 1000;
    jointPro_grass = zeros(n_grass, C);
    for i = 1 : EMLimit
        for j = 1 : C
            jointPro_grass(:, j) = mvnpdf(TrainsampleDCT_BG, mu_grass(j, :), sigma_grass(:, :, j)) * pi_grass(j); % mvnpdf returns the pdf of given X, mu, sigma
        end
        sumRow = sum(jointPro_grass, 2);
        likelihood_grass = sum(log(sumRow));
        hij_grass = jointPro_grass ./ sumRow;
        pi_grass = sum(hij_grass) / n_grass;
        mu_grass = (hij_grass' * TrainsampleDCT_BG) ./ sum(hij_grass)';
        for j = 1:C
            sigma_grass(:,:,j) = diag(diag(((TrainsampleDCT_BG - mu_grass(j,:))'.*hij_grass(:,j)'* ...
              (TrainsampleDCT_BG - mu_grass(j,:)) ./ sum(hij_grass(:,j),1)) + 0.0000001));
        end
        if i > 1
            if abs(likelihood_grass - likelihood_grass_previous) < 0.001
                break;
            end
```

```
        end
        likelihood_grass_previous = likelihood_grass;
    end
end

% BDR
function errorRate = ComputeBDR(zigzagImg, cheetah_mask, mu_cheetah, sigma_cheetah, pi_cheetah, mu_grass, sigma_grass, pi_grass)
    m = 248;
    n = 263;
    dimensionList = [1,2,4,8,16,24,32,40,48,56,64];
    lenList = length(dimensionList);
    errorMat = zeros(1, lenList);

    for curDim = 1 : lenList

        Kth = dimensionList(curDim);

        maskVec = zeros(m * n, 1);   % a vector of our mask, will resize it later

        for x = 1 : length(zigzagImg)

            pro_cheetah = 0;
            pro_grass   = 0;

            for y = 1:size(mu_cheetah,1)
                pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x, 1 : Kth), mu_cheetah(y, 1 : Kth), sigma_cheetah(1 : Kth, 1 : Kth,y)) * pi_cheetah(y);
            end

            % compute total BDR for grass
            for y = 1:size(mu_grass,1)
                pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1 : Kth), mu_grass(y, 1 : Kth),sigma_grass(1 : Kth, 1 : Kth,y))*pi_grass(y);
```

```
                end

                % decide whether the pixel is cheetah or grass
                if pro_cheetah > pro_grass
                    maskVec(x) = 1;
                end
            end

            maskMat = Vec2Mat(maskVec);

            errorMat(curDim) = Err(cheetah_mask, maskMat);
            errorRate = errorMat ./ (m * n);
        end
end

% convert a vector into a matrix
function mat = Vec2Mat(vec)
    m = 255 - 7;
    n = 270 - 7;
    mat = zeros(m,n);
    for x = 1:m
        mat(x,:) = vec(((x-1)*(n)+1):x*(n))';
    end
end

% compute the error rate
function errorRate = Err(idealMat, ansMat)
    errorRate = 0;
    [m, n] = size(ansMat);
    for i = 1 : m
        parfor j = 1 : n
            if(ansMat(i, j) ~= idealMat(i, j))
                errorRate = errorRate + 1;
            end
        end
    end
end
```

Code for Part B

```
clear
clc

% load all the data
load("ZigZagVec.mat")    % 1 x 64 vector
load("cheetahMat.txt")  % our image
load("TrainingSamplesDCT_8_new.mat")     % the training data sets
load("zigzagImg.mat") % the preprocessed image
load("cheetah_mask.mat")     % the preprocessed mask

% dimensions for the training & the parameters for EM
C = 8; % number of components
dimensionList = [1,2,4,8,16,24,32,40,48,56,64]; % number of dimensions of the space

% # of samples
[n_cheetah, ~] = size(TrainsampleDCT_FG);
[n_grass, ~] = size(TrainsampleDCT_BG);
totalSamples = (n_cheetah + n_grass);

% get the size of the image
[m, n] = size(cheetahMat);

% create a matrix
m = m - 7;
n = n - 7;
totalPixels = m * n;

% --- EM for cheetah ---

% initialize pi by creating a random normalized 1 x C matrix
pi_cheetah = randi(1, C);
pi_grass   = randi(1, C);
pi_cheetah = pi_cheetah / sum(pi_cheetah);
pi_grass   = pi_grass   / sum(pi_grass);

```

```
% initialize mu by choosing C samples randomly from the training data set
p = randperm(n_cheetah, C);     % pick C random numbers from 0 to the size of the sample of cheetah
mu_cheetah = zeros(C, 64);  % create an array for mu
mu_grass   = zeros(C, 64);  % create an array for mu
for i = 1 : C
    mu_cheetah(i, :) = TrainsampleDCT_FG(p(i), 64);
    mu_grass(i, :)   = TrainsampleDCT_BG(p(i), 64);
end

% initialize sigma by creating a diagonal matrix with random variable
sigma_cheetah = zeros(64, 64, C);
sigma_grass   = zeros(64, 64, C);
for i = 1 : C
    randomVec = rand(1, 64);     % generate a random vector (1 x 64)
    sigma_cheetah(:, :, i) = diag(randomVec);   % diagonlize the random vector
    randomVec = rand(1, 64);     % generate a random vector (1 x 64)
    sigma_grass(:, :, i) = diag(randomVec);   % diagonlize the random vector
end

EMLimit = 1000; % maximum iterations

% ---- Start of EM for cheetah ----
jointPro_cheetah = zeros(n_cheetah, C);
for i = 1 : EMLimit
    % E step
    for j = 1 : C
        % P_{X, Z}(x, z; psi) = P_{X|Z}(x|z; psi) * P_Z(z; psi)
        jointPro_cheetah(:, j) = mvnpdf(TrainsampleDCT_FG, mu_cheetah(j, :), sigma_cheetah(:, :, j)) * pi_cheetah(j); % mvnpdf returns the pdf of given X, mu, sigma
```

```
    end

    % calculate the log likelihood of the
    sumRow = sum(jointPro_cheetah, 2);    % returns the sum of the elements in each row
    likelihood_cheetah = sum(log(sumRow));

    % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j, sigma_j) * pi_j)
    hij_cheetah = jointPro_cheetah ./ sumRow;

    % M step

    % update pi for (n+1)
    % pi = 1 / n * (sum(hij))
    pi_cheetah = sum(hij_cheetah) / n_cheetah;

    % update mu for (n+1)
    % mu = sum(hij * x)/sum(hij)
    % TODO: mu_cheetah changes dimension
    mu_cheetah = (hij_cheetah' * TrainsampleDCT_FG) ./ sum(hij_cheetah)';

    % update sigma for (n+1)
    % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
    for j = 1:C
        sigma_cheetah(:,:,j) = diag(diag(((TrainsampleDCT_FG - mu_cheetah(j,:))'.*hij_cheetah(:,j)'* ...
            (TrainsampleDCT_FG - mu_cheetah(j,:)) ./ sum(hij_cheetah(:,j),1)) + 0.0000001));
    end

    % break condition
    % breaks the loop if the likelihood does not change more than 0.001
    if i > 1
        if abs(likelihood_cheetah - likelihood_cheetah_previous) < 0.001
            break;
```

```
        end
    end

    % store the current result as the previous
    likelihood_cheetah_previous = likelihood_cheetah;

end
% —— End of EM for cheetah ——

% —— Start of EM for grass ——
jointPro = zeros(n_grass, C);
for i = 1 : EMLimit
    % E step
    for j = 1 : C
        % P_{X, Z}(x, z; psi) = P_{X|Z}(x|z; psi) * P_Z(z; psi)
        jointPro(:, j) = mvnpdf(TrainsampleDCT_BG, mu_grass(j, :), sigma_grass(:, :, j)) * pi_grass(j); % mvnpdf returns the pdf of given X, mu, sigma
    end

    % calculate the log likelihood of the
    sumRow = sum(jointPro, 2);    % returns the sum of the elements in each row
    likelihood_grass = sum(log(sumRow));

    % h_ij = (G(x_i, u_j, sigma_j) * pi_j) / sum(G(x_i, u_j, sigma_j) * pi_j)
    hij_grass = jointPro ./ sumRow;

    % M step

    % update pi for (n+1)
    % pi = 1 / n * (sum(hij))
    pi_grass = sum(hij_grass) / n_grass;

    % update mu for (n+1)
    % mu = sum(hij * x)/sum(hij)
    mu_grass = hij_grass' * TrainsampleDCT_BG ./ sum(hij_grass
```

```
        )';

    % update sigma for (n+1)
    % sigma^2 = sum(hij * (x - mu)^2) / sum(hij)
    for j = 1:C
        sigma_grass(:, :, j) = diag(diag(((TrainsampleDCT_BG - mu_grass(j,:))'.*hij_grass(:,j)'* ...
            (TrainsampleDCT_BG - mu_grass(j,:)) ./ sum(hij_grass(:,j),1)) + 0.0000001));
    end

    % break condition
    % breaks the loop if the likelihood does not change more than 0.001
    if i > 1
        if abs(likelihood_grass - likelihood_grass_previous) < 0.001
            break;
        end
    end

    % store the current result as the previous
    likelihood_grass_previous = likelihood_grass;

end
% ---- End of EM for grass ----

% ---- BDR ----
lenList = length(dimensionList);
errorMat = zeros(1, lenList);
for curDim = 1 : lenList

    Kth = dimensionList(curDim);

    % compare BDR for EM
    maskVec = zeros(m * n, 1);    % a vector of our mask, will resize it later

    for x = 1 : length(zigzagImg)
```

```

        % set the probability of each class as zero
        pro_cheetah = 0;
        pro_grass   = 0;

        % compute total BDR for cheetah
        for y = 1:size(mu_cheetah,1)
            pro_cheetah = pro_cheetah + mvnpdf(zigzagImg(x, 1
                : Kth), mu_cheetah(y, 1 : Kth), sigma_cheetah(1
                : Kth, 1 : Kth,y)) * pi_cheetah(y);
        end

        % compute total BDR for grass
        for y = 1:size(mu_grass,1)
            pro_grass = pro_grass + mvnpdf(zigzagImg(x, 1 :
               Kth), mu_grass(y, 1 : Kth),sigma_grass(1 : Kth,
               1 : Kth,y))*pi_grass(y);
        end

        % decide whether the pixel is cheetah or grass
        if pro_cheetah > pro_grass
            maskVec(x) = 1;
        end
    end

    % resize the vector to matrix
    maskMat = Vec2Mat(maskVec);

    % compute the error rate
    errorMat(curDim) = Err(cheetah_mask, maskMat);
    errorRate = errorMat ./ (m * n);
end

% -- functions --

% convert the vector into the matrix
function mat = Vec2Mat(vec)
    m = 255 - 7;
    n = 270 - 7;
```

```
        mat = zeros(m,n);
        for x = 1:m
            mat(x,:) = vec(((x-1)*(n)+1):x*(n))';
        end
end

% compute the error rate
function errorRate = Err(idealMat, ansMat)
        errorRate = 0;
        [m, n] = size(ansMat);
        for i = 1 : m
            parfor j = 1 : n
                if(ansMat(i, j) ~= idealMat(i, j))
                    errorRate = errorRate + 1;
                end
            end
        end
end
```