ECE 271A Homework Set Two (Quiz)

CHENG-YAN JUANG

(a)

By analyzing the training data TrainingSamplesDCT_8_new.mat provided, the Maximum Likelihood Estimates (MLE) of the prior possibilities are computed based on the formula derived in Problem 2 as shown in Figure1. The results are identical to the estimate in Hw1. This is because the intuitive formula (calculating class probabilities as the ratio of class sample counts to the total number of samples) in Hw1 is the same as applying the MLE principle to a multinomial distribution. Therefore, what we did in Hw1 was the MLE.

Formula in Hw1:

$$Py(cheetah) = \frac{N\_FG}{N\_FG + N\_BG}, Py(grass) = 1 - Py(cheetah)$$

Formula derived in Problem 2 (Figure1):

$$Py(cheetah) = \frac{N\_FG}{N\_FG + N\_BG}, Py(grass) = \frac{B\_FG}{N\_FG + N\_BG}$$

(a) MLE

(1) choose a parametric model $P_{X|Y}(x|i;\theta_i)$

$P_X(k;\pi) = \pi_k$, $\pi = [\pi_1, \cdots, \pi_K]$

(2) assemble collection of datasets $D^{(i)} = \{x_1^{(i)}, \cdots x_n^{(i)}\}$

n observations $\Rightarrow C = (C_1, \cdots, C_N)$

(3) Find MLE $P_X(D;\theta)$

$$P_X(C;\pi) = \frac{n!}{\prod_{k=1}^N C_k!} \prod_{k=1}^N \pi_k^{C_k} = \frac{n!}{C_1! \times C_2! \cdots C_N!} \times (\pi_1^{C_1} \times \pi_2^{C_2} \times \cdots \pi_N^{C_N}$$

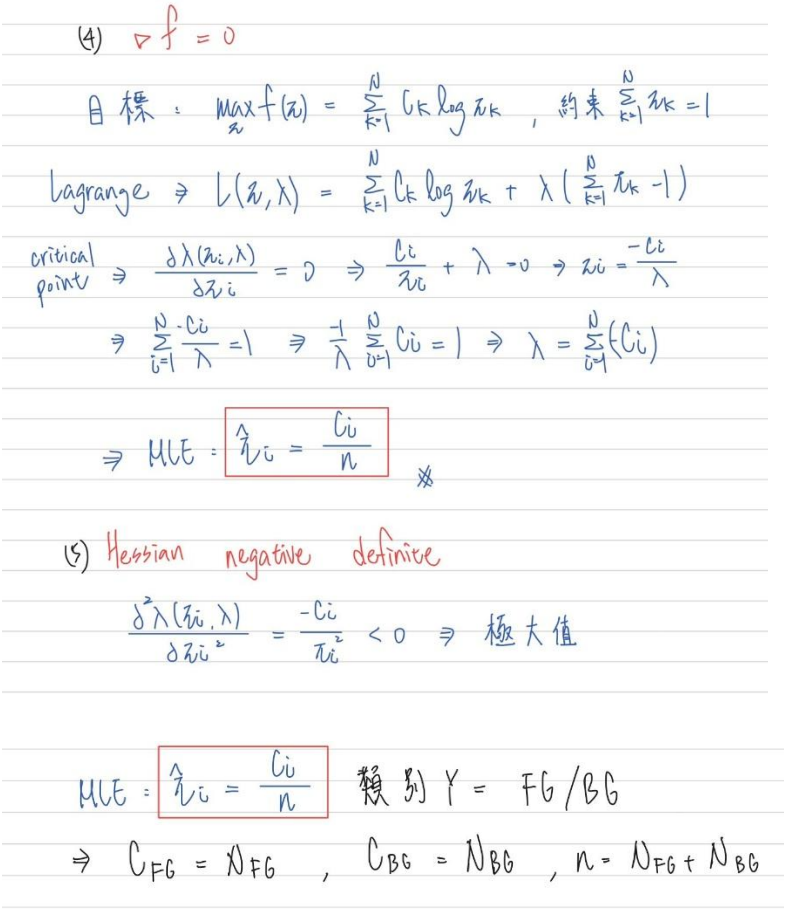
$$\Rightarrow l(\pi) = \log P_X(C;\pi) = \log n! - \log\left(\prod_{k=1}^N C_k!\right) + \log\left(\prod_{k=1}^N \pi_k^{C_k}\right)$$

$$= \log n! - \sum_{k=1}^N \log C_k! + \sum_{k=1}^N \log(\pi_k^{C_k})$$

$$= \left(\log n! - \sum_{k=1}^N \log C_k!\right) + \sum_{k=1}^N C_k \log \pi_k$$

const

$$\Rightarrow l(\pi) = \text{const} + \sum_{k=1}^N C_k \log \pi_k$$



We obtained:

- **Prior Probability of Cheetah (Foreground): 0.1919**
- **Prior Probability of Grass (Background): 0.8081**

(b)

For each of the 64 DCT coefficients, I estimated the class-conditional Gaussian parameters (mean and covariance) for both foreground and background using MLE:

$$\mu_i = \text{mean}(x_i), \quad \sigma_i^2 = \text{var}(x_i)$$

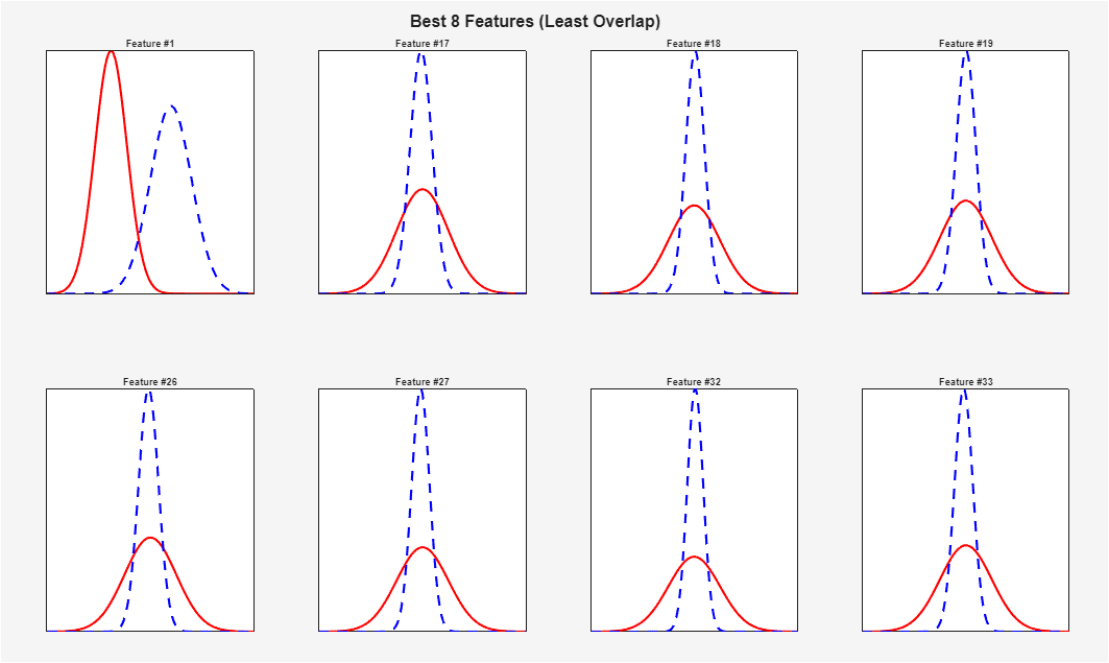
where x_i denotes the i-th DCT feature across all training samples.

Then, for each feature dimension, I plotted two 1-D Gaussian curves:

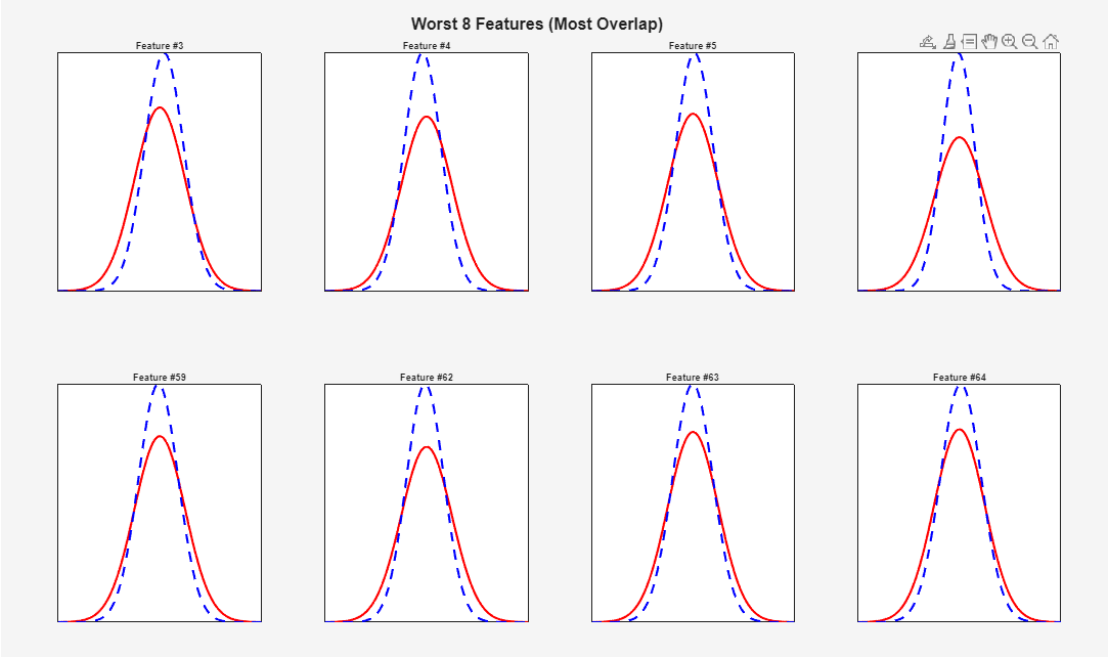
- Red solid line: cheetah distribution $\mathcal{N}(\mu_{FG}, \sigma_{FG}^2)$
- Blue dashed line: grass distribution $\mathcal{N}(\mu_{BG}, \sigma_{BG}^2)$

I used MATLAB's subplot to visualize all 64 features in one figure, then from visual inspection, I selected:

- **Best 8 features**: #1, #17, #18, #19, #26, #27, #32, #33



- **Worst 8 features:** #3, #4, #5, #6, #59, #62, #63, #64



(c)

A sliding 8×8 window (stride = 1) was used to compute DCT coefficients for every block. Each block's coefficients were rearranged into zig-zag order using Zig-Zag Pattern.txt, forming a 64-D feature vector.

For each block, I computed the log-likelihood under the multivariate Gaussian assumption for both classes (cheetah and grass) using the formula:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - \frac{1}{2}\log\ |\Sigma_i| + \log\ P(Y=i)$$

The Bayesian decision rule was then applied by comparing the discriminant scores, the block was labeled as *cheetah* if $g_{FG}(x) > g_{BG}(x)$, and *grass* otherwise. This procedure was implemented for both the full 64-D model and the reduced Best-8D model. The result of this section is two logical masks, A64 and A8, as shown in Figure 2.

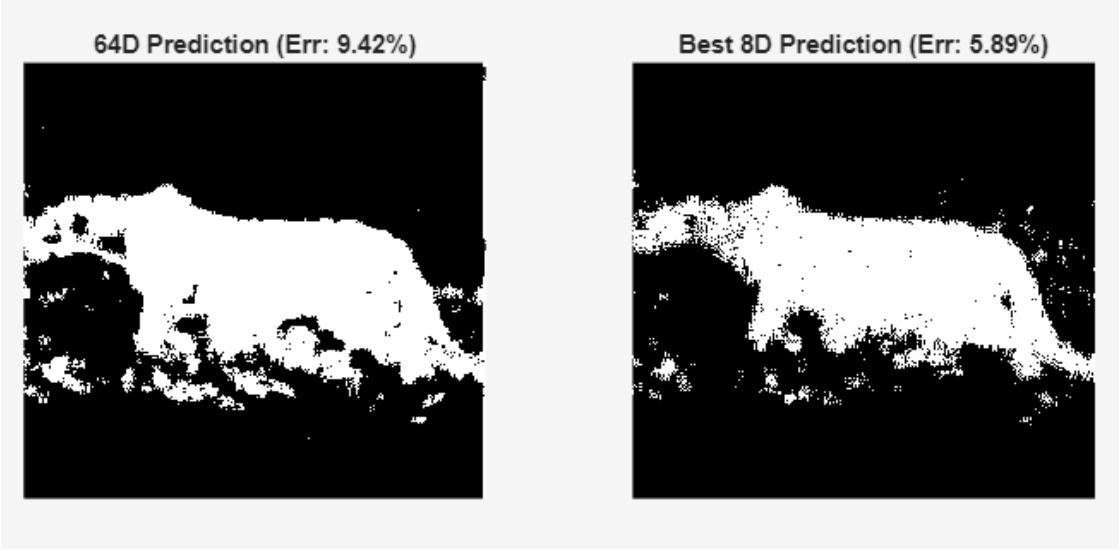


Figure 2

The error rate is computed by finding the proportion of pixels where the prediction mask does not equal the ground truth mask.

- **i) Probability of error for 64-dimensional: 9.42%**
- **ii) Probability of error for 8-dimensional: 5.89%**
- **Explanation:**

The 8-dimensional classifier's error rate is lower than the 64-dimensional classifier's, even though the 64D model has more information. With a limited number of training

samples, this high-dimensional estimation is very noisy and unstable, leading to a model that is overfitted to the training data and generalizes poorly. Because the model is trying to fit a complex, high-dimensional Gaussian to limited data, it ends up capturing random noise or sample-specific variations rather than the true underlying pattern. This means it performs well on the training data but generalizes poorly to unseen test images, a case of overfitting. The 8D model on the other hand is far simpler. Because it only uses the most discriminative features selected, it captures the most important aspects of the data while being far more robust and less prone to overfitting, it captures the essential information with fewer, cleaner features,
it avoids overfitting and therefore generalizes better, resulting in a lower classification error rate on the test image.

Table of Contents

a) 1
b) 1
c) 4
error 6

a)

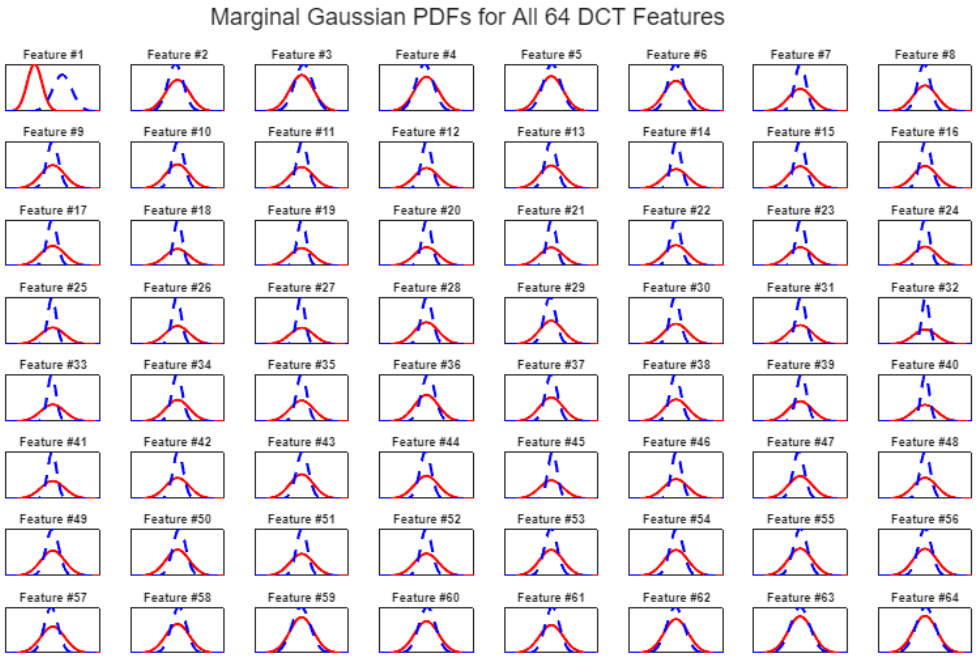
```
load('TrainingSamplesDCT_8_new.mat')
N_FG = size(TrainsampleDCT_FG);
N_BG = size(TrainsampleDCT_BG);
N = N_FG(1) + N_BG(1);
Py_cheetah = N_FG(1)/ N;
Py_grass = N_BG(1)/ N;
```

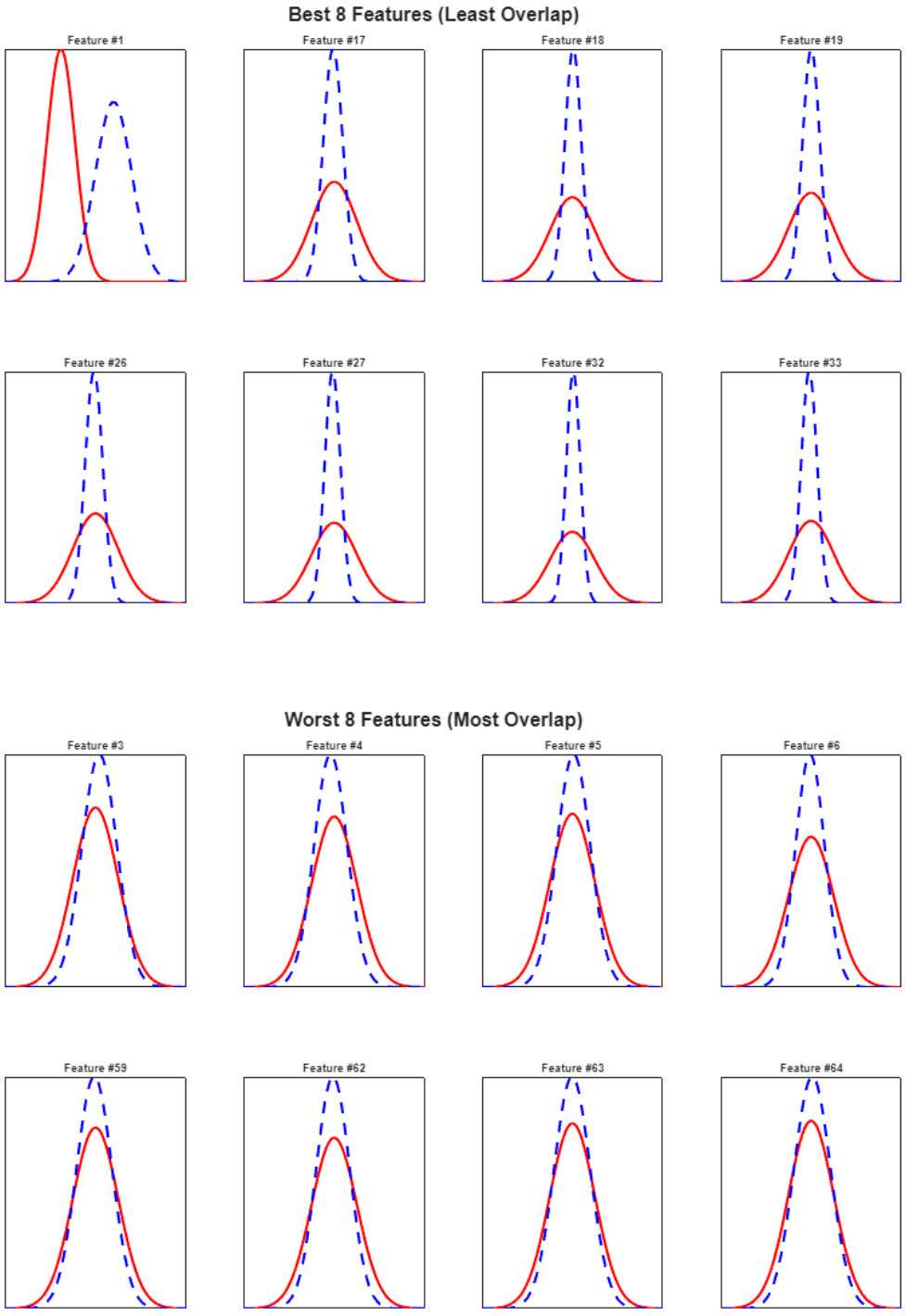
b)

```
% 1)
u_FG  = mean(TrainsampleDCT_FG, 1);
u_BG  = mean(TrainsampleDCT_BG, 1);

cov_FG = cov(TrainsampleDCT_FG, 1);
cov_BG = cov(TrainsampleDCT_BG, 1);

var_FG = diag(cov_FG)';
var_BG = diag(cov_BG)';

% 2)
figure(1); clf;
sgtitle('Marginal Gaussian PDFs for All 64 DCT Features');
for k = 1:64
    subplot(8,8,k);
    hold on; grid on; box on;

    sd_Fx = sqrt(var_FG(k));
    sd_Bx = sqrt(var_BG(k));
    u_Fx = u_FG(k);
    u_Bx = u_BG(k);

    lo_bound = min([u_Fx-4*sd_Fx, u_Bx-4*sd_Bx]);
    hi_bound = max([u_Fx+4*sd_Fx, u_Bx+4*sd_Bx]);
    x  = linspace(lo_bound, hi_bound, 300);

    G_Fx = normpdf(x, u_Fx, sd_Fx);
    G_Bx = normpdf(x, u_Bx, sd_Bx);

    plot(x, G_Fx, 'r-',  'LineWidth', 1.6);
    plot(x, G_Bx, 'b--', 'LineWidth', 1.6);
```

```
    title(sprintf('Feature #%d', k), 'FontSize', 7, 'FontWeight', 'bold');
    set(gca, 'XTick', [], 'YTick', []);
    axis tight;

end

best_8  = [ 1 17 18 19 26 27 32 33 ];
worst_8 = [ 3 4 5 6 59 62 63 64 ];

% 3)
figure(2); clf;
sgtitle('Best 8 Features (Least Overlap)', 'FontWeight','bold', 'FontSize',
12);
for i = 1:8
    k = best_8(i);
    subplot(2,4,i);
    hold on; grid on; box on;

    sd_Fx = sqrt(var_FG(k));
    sd_Bx = sqrt(var_BG(k));
    u_Fx = u_FG(k);
    u_Bx = u_BG(k);

    lo_bound = min([u_Fx-4*sd_Fx, u_Bx-4*sd_Bx]);
    hi_bound = max([u_Fx+4*sd_Fx, u_Bx+4*sd_Bx]);
    x  = linspace(lo_bound, hi_bound, 300);

    G_Fx = normpdf(x, u_Fx, sd_Fx);
    G_Bx = normpdf(x, u_Bx, sd_Bx);

    plot(x, G_Fx, 'r-',  'LineWidth', 1.6);
    plot(x, G_Bx, 'b--', 'LineWidth', 1.6);

    title(sprintf('Feature #%d', k), 'FontSize', 7, 'FontWeight', 'bold');
    set(gca, 'XTick', [], 'YTick', []);
    axis tight;

end

figure(3); clf;
sgtitle('Worst 8 Features (Most Overlap)', 'FontWeight','bold', 'FontSize',
12);
for i = 1:8
    k = worst_8(i);
    subplot(2,4,i);
    hold on; grid on; box on;

    sd_Fx = sqrt(var_FG(k));
    sd_Bx = sqrt(var_BG(k));
    u_Fx = u_FG(k);
    u_Bx = u_BG(k);

    lo_bound = min([u_Fx-4*sd_Fx, u_Bx-4*sd_Bx]);
    hi_bound = max([u_Fx+4*sd_Fx, u_Bx+4*sd_Bx]);
```

```
    x  = linspace(lo_bound, hi_bound, 300);

    G_Fx = normpdf(x, u_Fx, sd_Fx);
    G_Bx = normpdf(x, u_Bx, sd_Bx);

    plot(x, G_Fx, 'r-',  'LineWidth', 1.6);
    plot(x, G_Bx, 'b--', 'LineWidth', 1.6);

    title(sprintf('Feature #%d', k), 'FontSize', 7, 'FontWeight', 'bold');
    set(gca, 'XTick', [], 'YTick', []);
    axis tight;

end
```



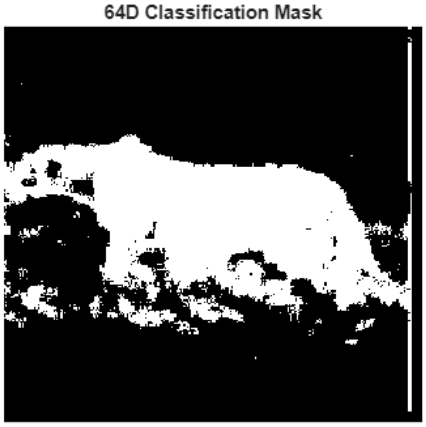


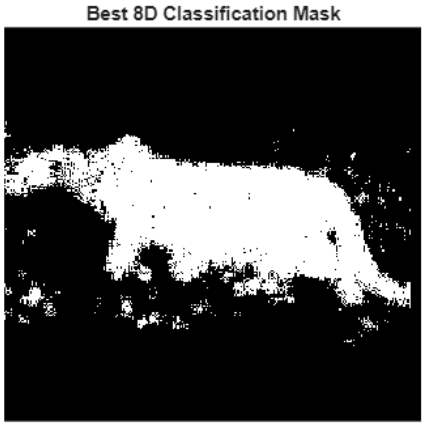
c)

```
test_img = im2double(imread('cheetah.bmp'));
[H, W] = size(test_img);
```

```
logPy_cheetah = log(Py_cheetah);
logPy_grass = log(Py_grass);

inv_FG  = inv(cov_FG);
inv_BG  = inv(cov_BG);
log_detFG = log(det(cov_FG));
log_detBG = log(det(cov_BG));


u_FG8   = u_FG(best_8).';
u_BG8   = u_BG(best_8).';
cov_FG8 = cov_FG(best_8, best_8);
cov_BG8 = cov_BG(best_8, best_8);
inv_FG8 = inv(cov_FG8);
inv_BG8 = inv(cov_BG8);
log_detFG8 = log(det(cov_FG8) + realmin);
log_detBG8 = log(det(cov_BG8) + realmin);

A64 = false(H, W);
A8  = false(H, W);

% zig zag
zig_zag_array = load('Zig-Zag Pattern.txt');
zig_zag_array = zig_zag_array + 1;
order_coef = zeros(1, 64);
for r = 1:8
    for c = 1:8
        k = zig_zag_array(r,c);
        order_coef(k) = sub2ind([8,8], r, c);
    end
end

for i = 1:(H-7)
    for j = 1:(W-7)
        D = dct2(test_img(i:i+7, j:j+7));
        x64 = D(order_coef).';

        xuF = x64 - u_FG(:);
        xuB = x64 - u_BG(:);
        gFG = -0.5*(xuF.'*inv_FG*xuF) - 0.5*log_detFG + logPy_cheetah;
        gBG = -0.5*(xuB.'*inv_BG*xuB) - 0.5*log_detBG + logPy_grass;

        if gFG > gBG
            A64(i, j) = 1;
        else
            A64(i, j) = 0;
        end

        x8  = x64(best8);
        xuF8 = x8 - u_FG8;
        xuB8 = x8 - u_BG8;
        gFG8 = -0.5*(xuF8.'*inv_FG8*xuF8) - 0.5*log_detFG8 + logPy_cheetah;
        gBG8 = -0.5*(xuB8.'*inv_BG8*xuB8) - 0.5*log_detBG8 + logPy_grass;
```

```
        if gFG8 > gBG8
            A8(i, j) = 1;
        else
            A8(i, j) = 0;
        end
    end
end


figure;

subplot(1, 2, 1);
imshow(A64);
title('64D Classification Mask');

subplot(1, 2, 2);
imshow(A8);
title('Best 8D Classification Mask');
```



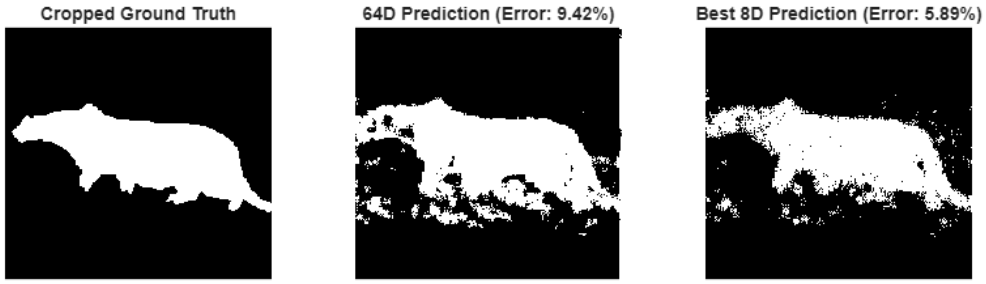


error

```
ground_truth = imread('cheetah_mask.bmp');
ground_truth = im2double(ground_truth);
ground_truth = ground_truth > 0.5;


rows = 1:(H-7);
cols = 1:(W-7);
```

```
ground_truth_eval = ground_truth(rows, cols);

A64_eval  = A64(rows, cols);
err_rate_64 = mean(A64_eval(:) ~= ground_truth_eval(:));

A8_eval  = A8(rows, cols);
err_rate_8 = mean(A8_eval(:) ~= ground_truth_eval(:));

fprintf('Error Rate (64D Classifier): %.4f (%.2f%%)\n', err_rate_64,
100*err_rate_64);
fprintf('Error Rate (Best 8D Classifier): %.4f (%.2f%%)\n', err_rate_8,
100*err_rate_8);

figure;
colormap(gray(255));

subplot(1, 3, 1);
imagesc(ground_truth_eval);
axis image off;
title('Cropped Ground Truth');

subplot(1, 3, 2);
imagesc(A64_eval);
axis image off;
title(sprintf('64D Prediction (Error: %.2f%%)', 100*err_rate_64));

subplot(1, 3, 3);
imagesc(A8_eval);
axis image off;
title(sprintf('Best 8D Prediction (Error: %.2f%%)', 100*err_rate_8));
```

```
Error Rate (64D Classifier): 0.0942 (9.42%)
Error Rate (Best 8D Classifier): 0.0589 (5.89%)
```



Published with MATLAB® R2025b