ECE 271A Homework Set One (Quiz)

CHENG-YAN JUANG

(a)

Answer:

By analyzing the training data TrainingSamplesDCT8.mat provided, the priors are computed based on the number of training samples for the foreground (FG) and background (BG) blocks, using this formula:

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

We obtained:

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

Code and Explanation:

The size function is used to count the total number of samples for each class, and the proportion of each class relative to the total number of samples N is calculated. This ratio serves as our estimate for the prior probabilities. The "Test" variable is just for checking.

```
%% a)
load('TrainingSamplesDCT_8.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;
Test = 1-Py_cheetah
```

(b)

Answer:

As shown in the figures below, the probability distribution histograms for both classes' feature index were successfully computed and plotted.

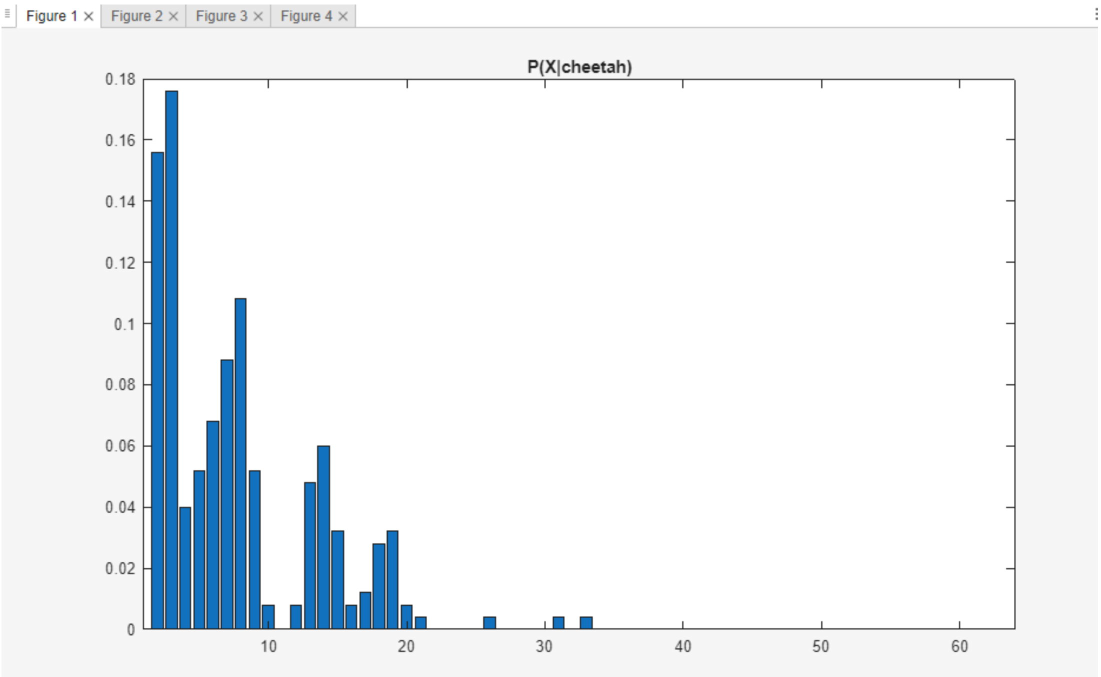


Figure 1. Probability of feature index given cheetah (Foreground)

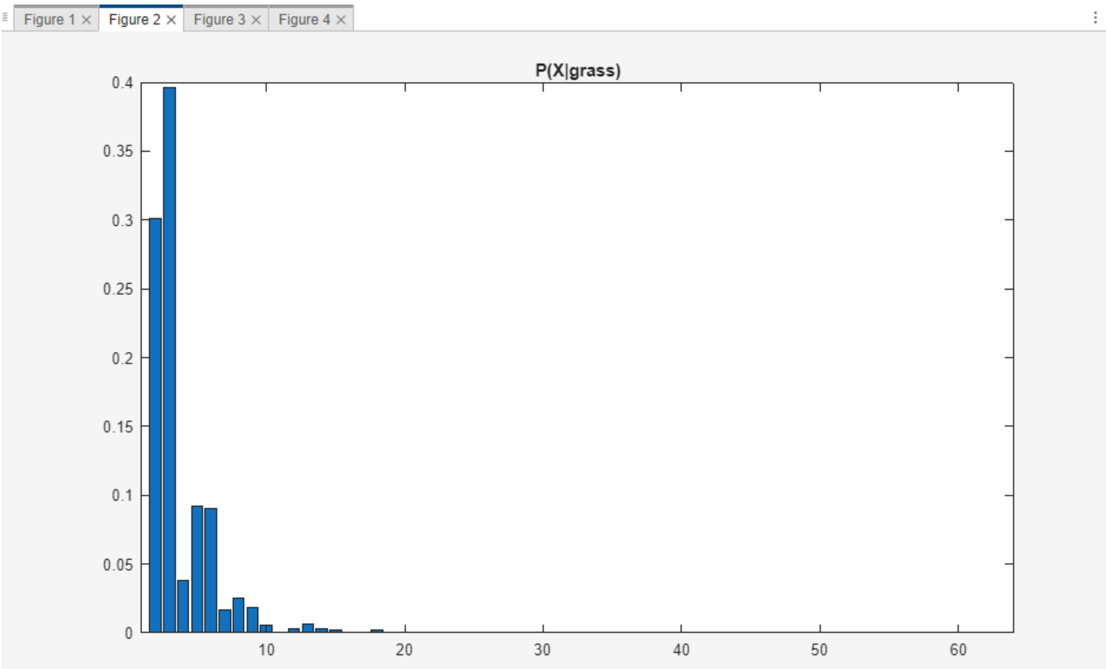


Figure 2. Probability of feature index given grass (Foreground)

<u>Code and Explanation:</u>

Each 8×8 DCT block is represented by a 64-dimensional feature vector.

For each training sample, we:

1. Compute the magnitude of each DCT coefficient “absVec”.

2. Identify the position of the second largest coefficient "X_BG(i)".

3. Record this position and build a histogram of occurrences.

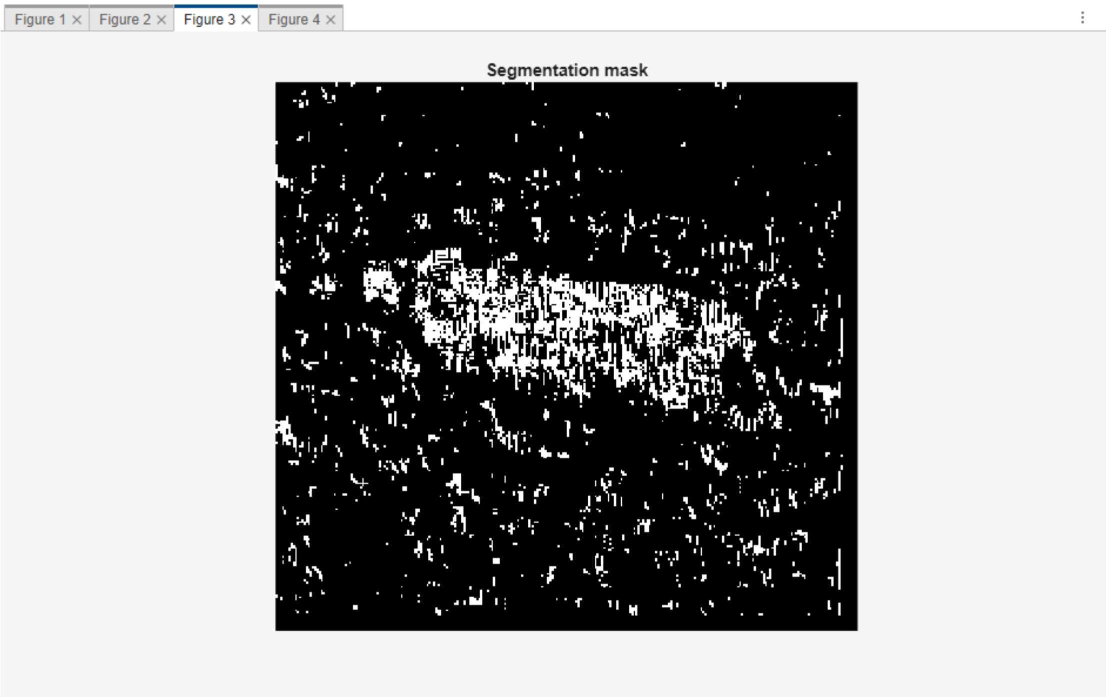
4. Normalize the histogram to obtain probability distribution, , yielding "P_X_given_FG" and "P_X_given_BG".

```
%% b)
data_FG = TrainsampleDCT_FG;
numSamples = size(data_FG, 1);
X_FG = zeros(numSamples, 1);
N_block_FG = N_FG(1);

data_BG = TrainsampleDCT_BG;
numSamples = size(data_BG, 1);
X_BG = zeros(numSamples, 1);
N_block_BG = N_BG(1);


for i = 1:N_block_FG

    vec = data_FG(i, :);
    absVec = abs(vec);
    [B, sortedIndex] = sort(absVec, 'descend');
    X_FG(i) = sortedIndex(2);
end


for i = 1:N_block_BG
    vec = data_BG(i, :);
    absVec = abs(vec);
    [B, sortedIndex] = sort(absVec, 'descend');
    X_BG(i) = sortedIndex(2);
end


edges = 0.5:1:64.5; bins = 1:64;
P_X_given_FG = histcounts(X_FG, edges, 'Normalization','probability');
P_X_given_BG = histcounts(X_BG, edges, 'Normalization','probability');

figure; bar(bins, P_X_given_FG); title('P(X|cheetah)'); xlim([1 64]);
figure; bar(bins, P_X_given_BG); title('P(X|grass)');   xlim([1 64]);
```

(c)

Answer:

The classification result is stored in matrix A and visualized as the binary mask shown below:



<u>Code and Explanation:</u>

1. Use an 8×8 sliding window (stride = 1 pixel).
2. Compute the DCT for each block.
3. Apply zig-zag ordering to get a 1×64 feature vector.
4. Find the index of the second largest coefficient (feature X).
5. Compare the two posterior probabilities and assign the class label.
6. Classify using the MAP rule:

Classify as cheetah if $P_{X|Y}(X \mid cheetah)P_Y(cheetah) > P_{X|Y}(X \mid grass)P_Y(grass)$

```matlab
%% c)

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

A = zeros(H, W);

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)
        block = test_img(i:i+7, j:j+7);
        DCT = dct2(block);
        vec = DCT(order_coef);

        absVec = abs(vec);
        [~, sortedIndex] = sort(absVec, 'descend');
        X = sortedIndex(2);

        likelihood_FG = P_X_given_FG(X);
        likelihood_BG = P_X_given_BG(X);

        score_FG = likelihood_FG * Py_cheetah;
        score_BG = likelihood_BG * Py_grass;

        if score_FG > score_BG
            A(i, j) = 1;        % 1 = cheetah
        else
            A(i, j) = 0;        % 0 = grass
        end
    end
end

figure; imagesc(A); axis image off; colormap(gray(255));
title('Segmentation mask');
```

(d)

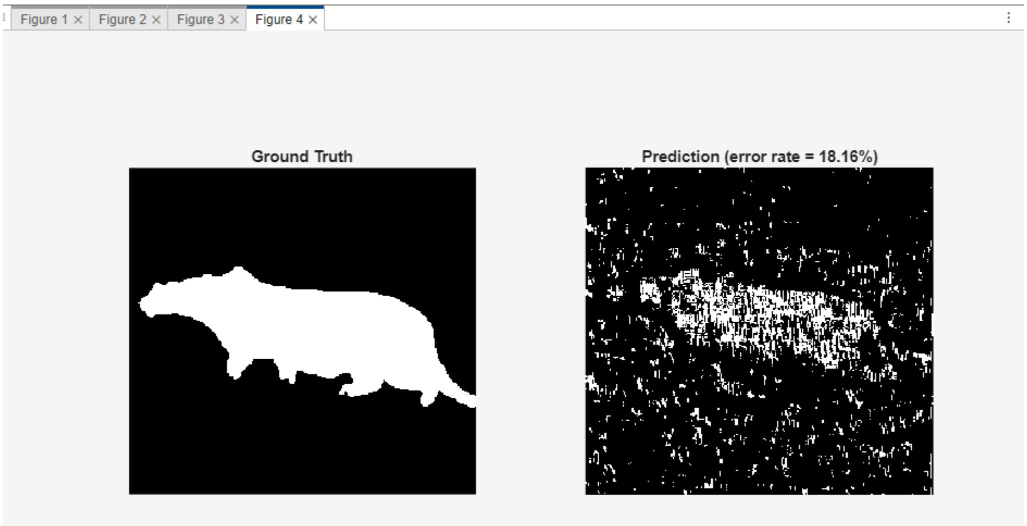
Answer:

By comparing our prediction mask with the ground truth, the calculated **probability of error is 0.1816 (18.16%)**. The figure below shows the ground truth (left) and our prediction (right) side-by-side.

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Hw1

Test =

    0.8081

Error rate = 0.1816 (18.16%)
```



<u>Code and Explanation:</u>

1. Convert the ground-truth mask to a binary format with values 0 (grass) and 1 (cheetah).
2. Crop both prediction and ground-truth masks to the same valid region.
3. Calculate the error rate using the 0/1 loss function.

```
%% d)

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

[H, W] = size(A);
rows = 1:(H-7);
cols = 1:(W-7);

A_eval  = A(rows, cols) > 0.5;
ground_truth_eval = ground_truth(rows, cols) > 0.5;

err_rate = mean(A_eval(:) ~= ground_truth_eval(:));

fprintf('Error rate = %.4f (%.2f%%)\n', err_rate, 100*err_rate);

figure;
subplot(1,2,1); imagesc(ground_truth_eval); axis image off; colormap(gray(255)); title('Ground Truth');
subplot(1,2,2); imagesc(A_eval);  axis image off; colormap(gray(255)); title(sprintf('Prediction (error rate = %.2f%%)', 100*err_rate));
```

Table of Contents

a)

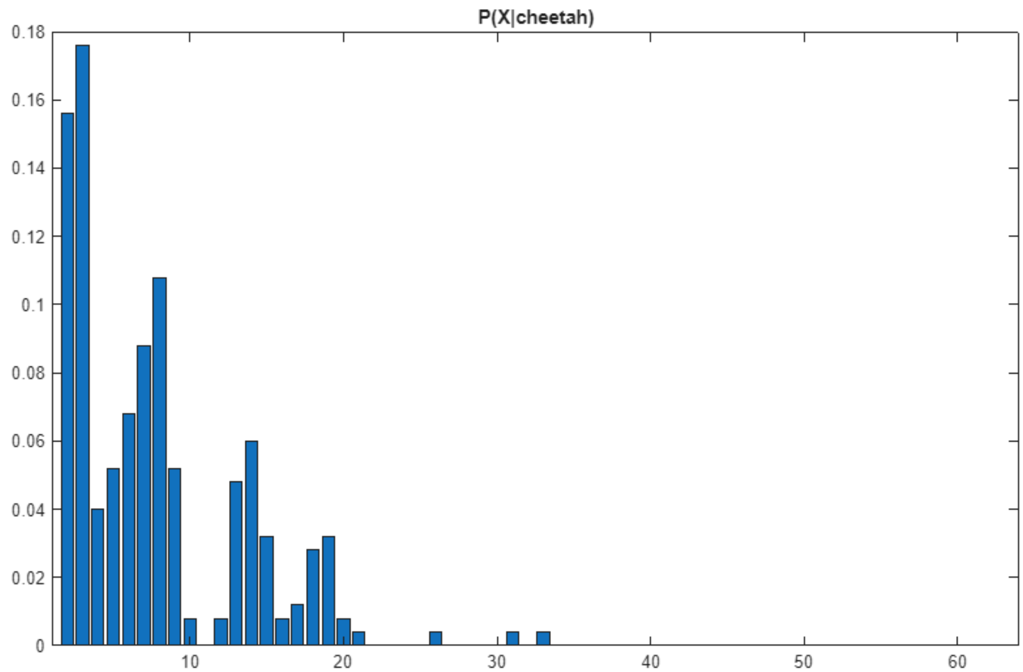
```
load('TrainingSamplesDCT_8.mat')
N_FG = size(TrainsampleDCT_FG);
N_BG = size(TrainsampleDCT_BG);
N = N_FG(1) + N_BG(1);
Py_cheetah = N_FG(1)/ N;
Py_grass = N_BG(1)/ N;
Test = 1-Py_cheetah
```

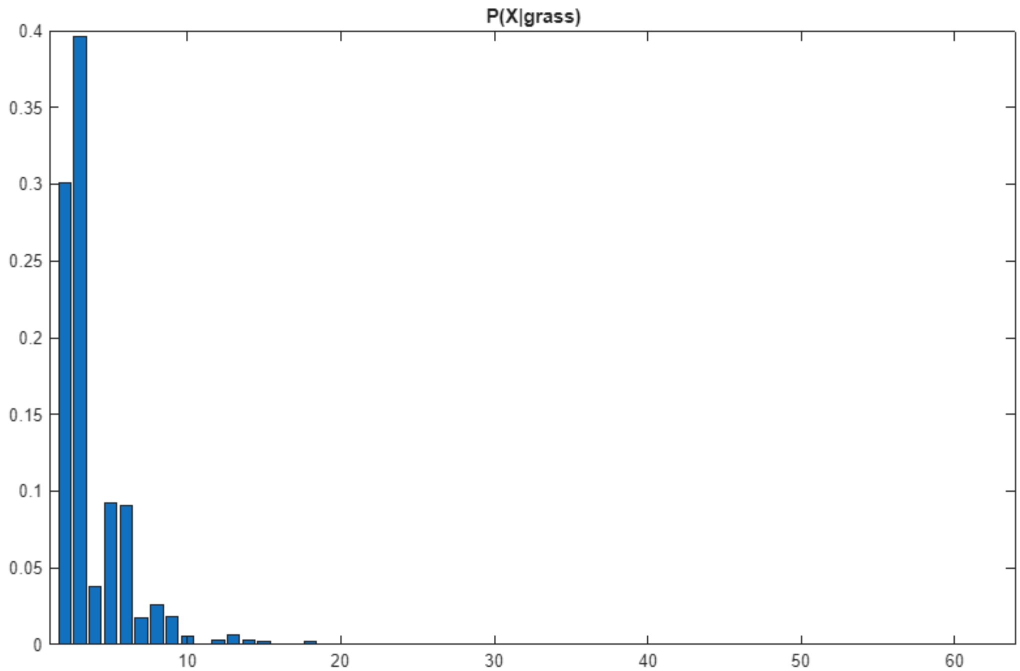
```
Test =

    0.8081
```

b)

```
data_FG = TrainsampleDCT_FG;
numSamples = size(data_FG, 1);
X_FG = zeros(numSamples, 1);
N_block_FG = N_FG(1);

data_BG = TrainsampleDCT_BG;
numSamples = size(data_BG, 1);
X_BG = zeros(numSamples, 1);
N_block_BG = N_BG(1);


for i = 1:N_block_FG
    vec = data_FG(i, :);
    absVec = abs(vec);
    [B, sortedIndex] = sort(absVec, 'descend');
    X_FG(i) = sortedIndex(2);
end


for i = 1:N_block_BG
    vec = data_BG(i, :);
    absVec = abs(vec);
    [B, sortedIndex] = sort(absVec, 'descend');
    X_BG(i) = sortedIndex(2);
end
```

```
edges = 0.5:1:64.5; bins = 1:64;
P_X_given_FG = histcounts(X_FG, edges, 'Normalization','probability');
P_X_given_BG = histcounts(X_BG, edges, 'Normalization','probability');

figure; bar(bins, P_X_given_FG); title('P(X|cheetah)'); xlim([1 64]);
figure; bar(bins, P_X_given_BG); title('P(X|grass)');   xlim([1 64]);
```

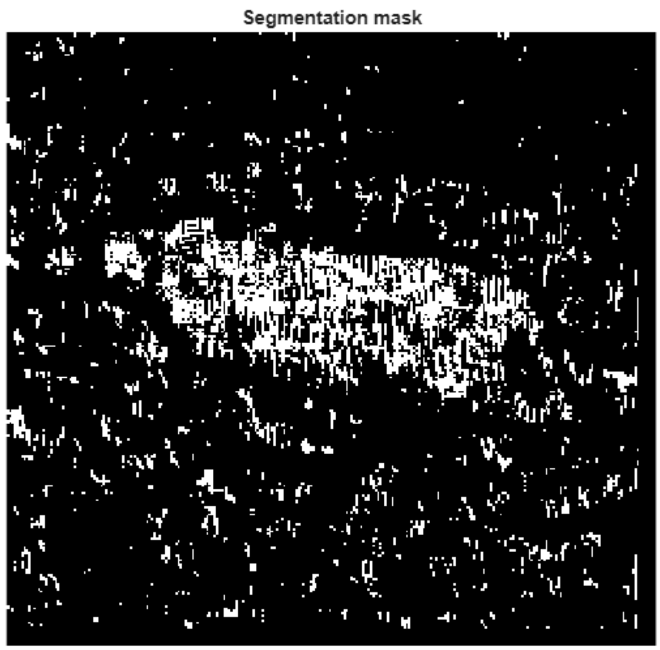




c)

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

A = zeros(H, W);

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)
        block = test_img(i:i+7, j:j+7);
        DCT = dct2(block);
        vec = DCT(order_coef);
        absVec = abs(vec);
        [~, sortedIndex] = sort(absVec, 'descend');
        X = sortedIndex(2);

        likelihood_FG = P_X_given_FG(X);
```

```
        likelihood_BG = P_X_given_BG(X);

        score_FG = likelihood_FG * Py_cheetah;
        score_BG = likelihood_BG * Py_grass;

        if score_FG > score_BG
            A(i, j) = 1;
        else
            A(i, j) = 0;
        end
    end
end

figure; imagesc(A); axis image off; colormap(gray(255));
title('Segmentation mask');
```

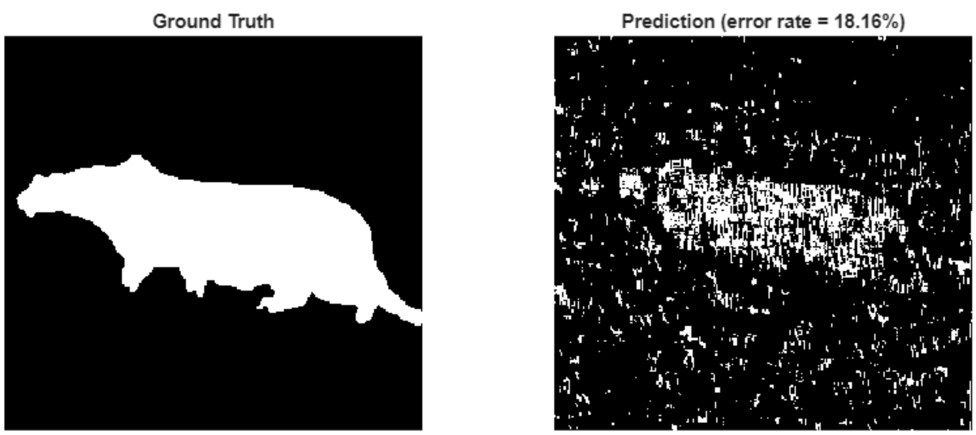


d)

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

[H, W] = size(A);
rows = 1:(H-7);
cols = 1:(W-7);

A_eval  = A(rows, cols) > 0.5;
ground_truth_eval = ground_truth(rows, cols) > 0.5;
```

```
err_rate = mean(A_eval(:) ~= ground_truth_eval(:));
fprintf('Error rate = %.4f (%.2f%%)\n', err_rate, 100*err_rate);


figure;
subplot(1,2,1); imagesc(ground_truth_eval); axis image off;
colormap(gray(255)); title('Ground Truth');
subplot(1,2,2); imagesc(A_eval);  axis image off; colormap(gray(255));
title(sprintf('Prediction (error rate = %.2f%%)', 100*err_rate));
```

```
Error rate = 0.1816 (18.16%)
```



Published with MATLAB® R2025b