ECE 271A HW1 Quiz

Chunlin Chen (PID: A59023021)

October 15, 2023

1 Prior Probabilities

We first load the training data in `TrainingSamplesDCT_8.mat` into Workspace, which contains two matrices `TrainsampleDCT_BG` and `TrainsampleDCT_FG`, with size of 1053×64 and 250×64 respectively. We know that each row of the matrix represents one 8×8 block in the image, so the blocks of background and foreground is 1053 and 250 respectively. Therefore, the prior probabilities are as follows:

$$P_Y(cheetah) = \frac{250}{1053 + 250} = 0.1919$$
$$P_Y(grass) = \frac{1053}{1053 + 250} = 0.8081$$

2 Index Histograms

For each row of a matrix, we find the index of its second largest element as the feature value of a block (ranges from 1 to 64), then we plot the histograms of feature values in background and foreground respectively, as shown in Figure 1 and Figure 2.

The likelihood for each feature value given class can also be easily computed, which is the number of blocks divided by the number of a certain feature value, as shown below.

$$P_{X|Y}(x|cheetah) = \frac{\text{number of times } x \text{ occurs}}{\text{number of cheetah blocks}}$$
$$P_{X|Y}(x|grass) = \frac{\text{number of times } x \text{ occurs}}{\text{number of grass blocks}}$$

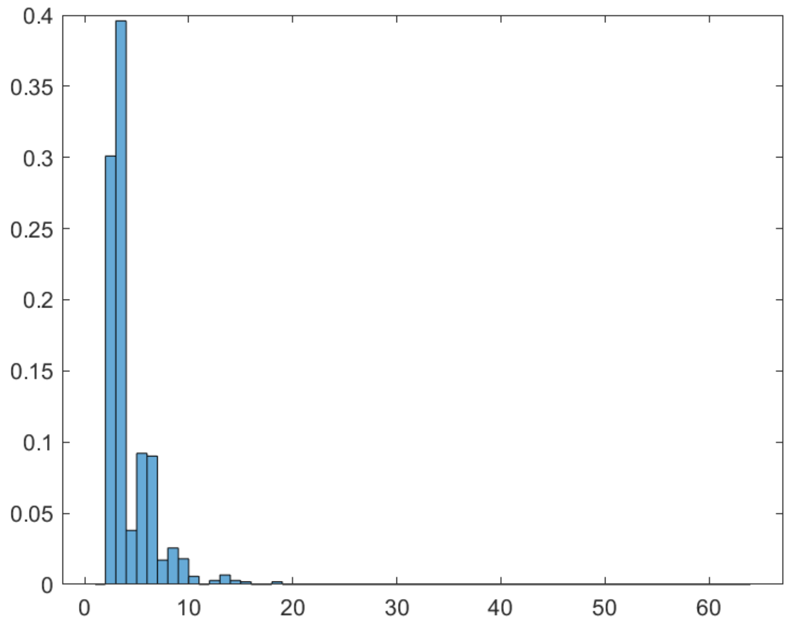

Figure 1: Histogram of Feature Values (1-64) in Background

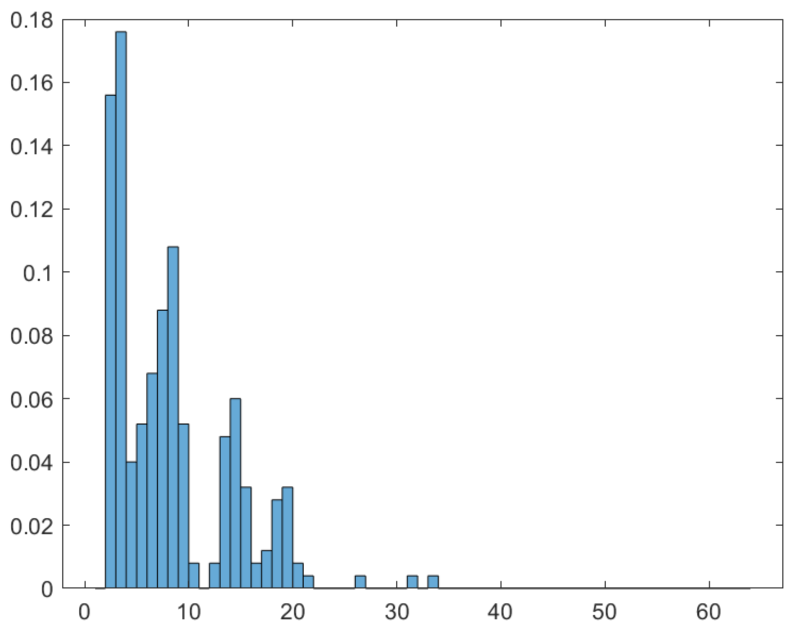

Figure 2: Histogram of Feature Values (1-64) in Foreground

3 Classification

We read the test image `cheetah.bmp` and conduct a DCT using a 8×8 block. To avoid blocky masks, we use a sliding window of 8×8 which moves by one pixel at each step to leverage the containing information in every pixel. In order to do so, we first apply zero padding to expand the image size to 262×277. The result of DCT is a 2040×2160 matrix or a 255×270 matrix made of 8×8 blocks. After taking an absolute value of the DCT coefficients matrix, we scan it using a Zig-Zag pattern to convert the 8×8 arrays of coefficients to 64D vectors. Then, we pick the position of the second largest magnitude as the feature value. Eventually, we can obtain a 255×270 matrix with each element representing the feature value of its corresponding block.

We use the minimum probability of error rule, which is the MAP rule, as shown below, to compute the state variable Y.

$$i^*(x) = \underset{i}{argmax}\, P_{X|Y}(x|i)P_Y(i), i \in \{\text{cheetah, grass}\}$$

We create a binary mask with 1's for foreground blocks and 0's for background blocks based on the predicted Y. The result is shown in Figure 3.

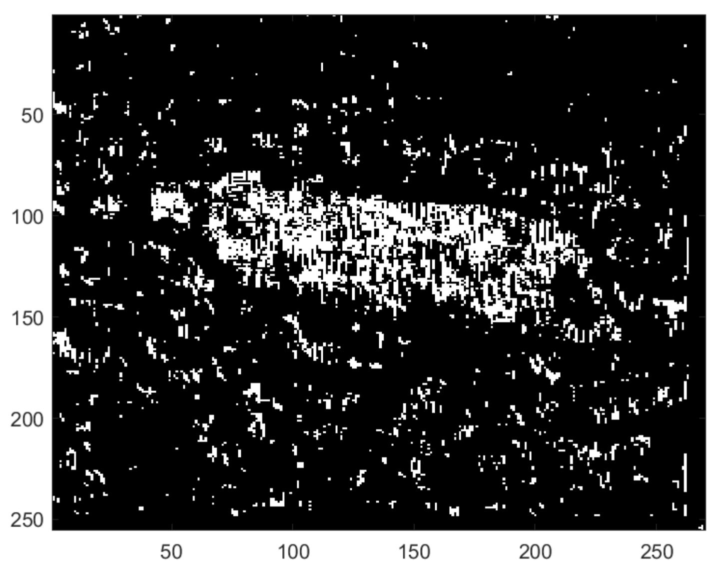

Figure 3: Prediction of MAP Rule

4 Probability of Errors

We load the ground truth image `cheetahmask.bmp` and compare it with our prediction.

The probability of error of our algorithm is shown below:

$$P(error) = \sum_{x \in X} P_{Y,X}(y \neq i^*(x), x) = \sum_{x \in X} \underset{y \in Y}{min}\, P_{X|Y}(x|y) P_Y(y)$$

And the result is $P(error) = 0.1527$. We use different methods to compute the probability, which can be found in the code.

5 Source Code

```
load("ECE 271A\homework1\TrainingSamplesDCT_8.mat")

% Compute priors
prob_bg = size(TrainsampleDCT_BG, 1) / (size(TrainsampleDCT_BG, 1) + size(TrainsampleDCT_FG, 1))
prob_fg = size(TrainsampleDCT_FG, 1) / (size(TrainsampleDCT_BG, 1) + size(TrainsampleDCT_FG, 1))

bg_train_indices = find2ndLargest(TrainsampleDCT_BG)
fg_train_indices = find2ndLargest(TrainsampleDCT_FG)

h_bg = histogram(bg_train_indices, 64, "BinEdges", [1:64], 'Normalization','probability')
h_fg = histogram(fg_train_indices, 64, "BinEdges", [1:64], 'Normalization','probability')

% Compute likelihood
prob_x_bg = histcounts(bg_train_indices, [1:65])
prob_x_bg = prob_x_bg / sum(prob_x_bg)
prob_x_fg = histcounts(fg_train_indices, [1:65])
prob_x_fg = prob_x_fg / sum(prob_x_fg)

```

```
ZigZagPattern = ZigZagPattern + 1
ZigZagPattern = int8(ZigZagPattern)

img = imread("ECE 271A\homework1\cheetah.bmp")
img = im2double(img)

% Zero Padding
right = zeros(255, 7);
bottom = zeros(7, 277);
img_pad = [[img right]; bottom]

% DCT
img_dct = dct_8(img, img_pad);
img_dct = abs(img_dct)

% ZigZag Scan
img_scan = blockproc(img_dct, [8 8], @(block_struct)
    ZigZagScan(block_struct.data, ZigZagPattern))
features = blockproc(img_scan, [1, 64], @(
   block_struct) find2ndLargest(block_struct.data));
features = int8(features)

% Create binary mask using BDR
mask = blockproc(features, [1, 1], @(block_struct)
   BDR(block_struct.data, prob_x_bg, prob_x_fg,
   prob_bg, prob_fg));
mask = int8(mask)
imagesc(mask)
colormap(gray(255))

ground_truth = imread("ECE 271A\homework1\
   cheetah_mask.bmp")
ground_truth = im2double(ground_truth)
imagesc(ground_truth)
colormap(gray(255))

ground_truth = int8(ground_truth)

```

```
% Compute probability of errors
% Method 1
p_error = 0;
diff = ground_truth - mask
diff_feature = features .* diff
false_fg = int8.empty;
false_bg = int8.empty;
x = (1:64);
for i = 1:size(diff_feature, 1)
    for j = 1:size(diff_feature, 2)
        if diff_feature(i, j) < 0 % Find false foreground (ground truth:0, predicted Y:1)
            false_fg = [false_fg -diff_feature(i, j)];
        end
        if diff_feature(i, j) > 0 % Find false background (ground truth:1, predicted Y:0)
            false_bg = [false_bg diff_feature(i, j)];
        end
    end
end
false_fg_x = intersect(x, false_fg)
false_bg_x = intersect(x, false_bg)
for i = 1:length(x)
    p_error = p_error + prob_x_bg(1, i) * prob_bg * ismember(i, false_fg_x) ...,
            + prob_x_fg(1, i) * prob_fg * ismember(i, false_bg_x);
end
p_error

% Method 2
p_error_1 = 0;
for i = 1:64
```

```matlab
    p_error_1 = p_error_1 + min(prob_x_bg(1, i) * prob_bg, prob_x_fg(1, i) * prob_fg);
end
p_error_1

% Method 3 (different result, equivalent to the proportion of wrongly predicted pixels in the whole image)
p_fg_gt = sum(sum(ground_truth==1)) / (size(ground_truth, 1) * size(ground_truth, 2));
p_bg_gt = sum(sum(ground_truth==0)) / (size(ground_truth, 1) * size(ground_truth, 2));
p_error_2 = sum(sum(diff==1)) / sum(sum(ground_truth==1)) * p_fg_gt + sum(sum(diff==-1)) / sum(sum(ground_truth==0)) * p_bg_gt
p_error_3 = 1 - sum(sum(mask==ground_truth)) / (size(ground_truth, 1) * size(ground_truth, 2)) % same as above

function vector = ZigZagScan(matrix, pattern)
    vector = zeros(1, size(matrix, 1) * size(matrix, 2));
    for i = 1:size(matrix, 1)
        for j = 1:size(matrix, 2)
            position = pattern(i, j);
            vector(1, position) = matrix(i, j);
        end
    end
end

function indices = find2ndLargest(sample)
    indices = zeros(1, size(sample, 1));
    for i = 1:size(sample, 1)
        [~, index] = maxk(sample(i, :), 2);
        indices(1, i) = indices(1, i) + index(1, 2);
    end
end
```

```
function mask = BDR(feature, P_x_bg, P_x_fg, P_bg, P_fg)
    if P_x_bg(1, feature) * P_bg >= P_x_fg(1, feature) * P_fg
        mask = 0;
    else
        mask = 1;
    end
end

function dct = dct_8(img, img_pad)
    dct = zeros(size(img, 1) * 8, size(img, 2) * 8);
    for i = 1:size(img, 1)
        for j = 1:size(img, 2)
            dct((8*i-7):(8*i), (8*j-7):(8*j)) = dct2(img_pad(i:i+7, j:j+7));
        end
    end
end
```