SYDE 575 Image Processing
Lab 5: Image Compression and Segmentation

Group 23: Laura Chambers, ID: [redacted]

1 Introduction

The purpose of this lab is to gain an understanding of image compression, colour segmentation, and neural networks.

In image compression, pixel information is discarded to reduce the amount of data that must be transmitted. The image compression process used in this lab includes multiple steps, including chroma subsampling, image transformation, and quantization.

Colour segmentation groups pixels together that share colour characteristics. K-means clustering is the method of colour segmentation explored in this lab.

Finally, neural networks are supervised machine learning algorithms that are used to recognize different tasks, such as classifying images. Neural networks are comprised of a model with parameters that are tuned via training to make accurate predictions.

The MATLAB code for this lab can be found in the Appendix.

2 Chroma Subsampling

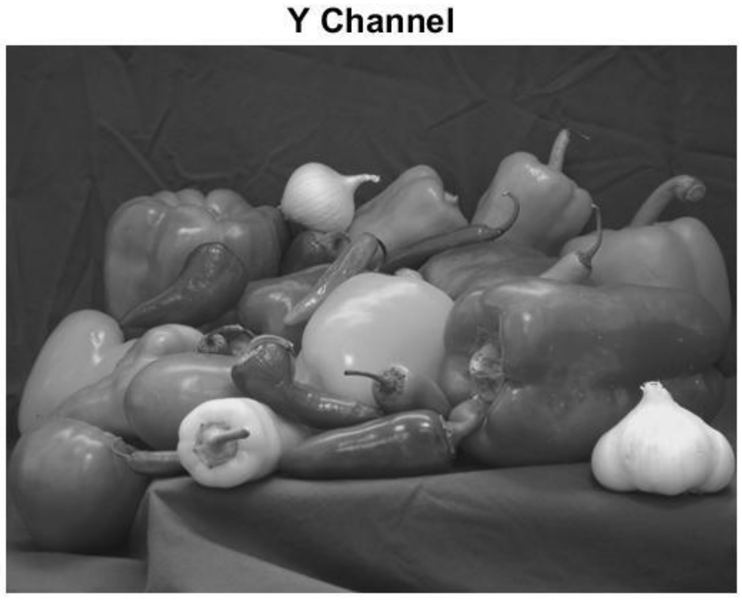


Figure 2-1 Y channel of YCbCr image

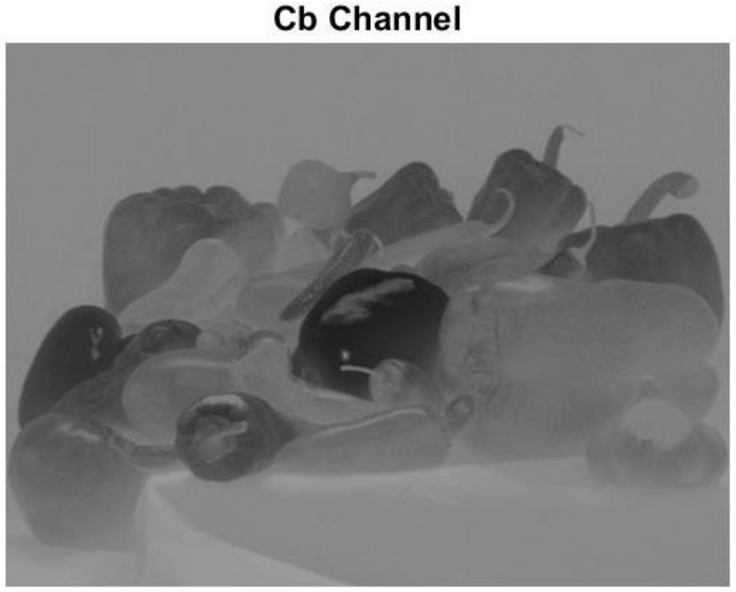


Figure 2-2 Cb channel of YCbCr image

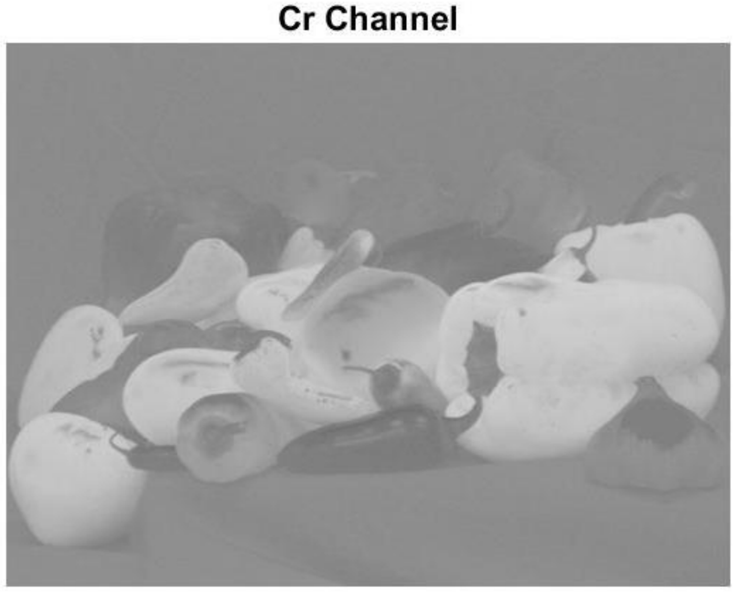


Figure 2-3 Cr channel of YCbCr image

1. In the YCbCr colour space, Y is luminance and Cb and Cr are colour difference components. Cb stands for chroma blue and represents the blue component of pixels minus luma (B-Y). Cr stands for chroma red and represents red minus luma (R-Y).

   The Cb and Cr channel images appear in grayscale. When the differences are plotted as a single channel image, each pixel value acts like a grayscale value. The difference is easiest to observe by looking at the large red pepper at the right side. The difference between red and Y is at its maximum for red pixels, therefore, they appear brightest (i.e., have the highest grayscale pixel value) in the Cr channel image in Figure 2-3.

2. The Y channel image contains much more detail than the Cb and Cr channel images. This suggests that intensity information is a more important signal for depicting fine details and edges compared to colour information.

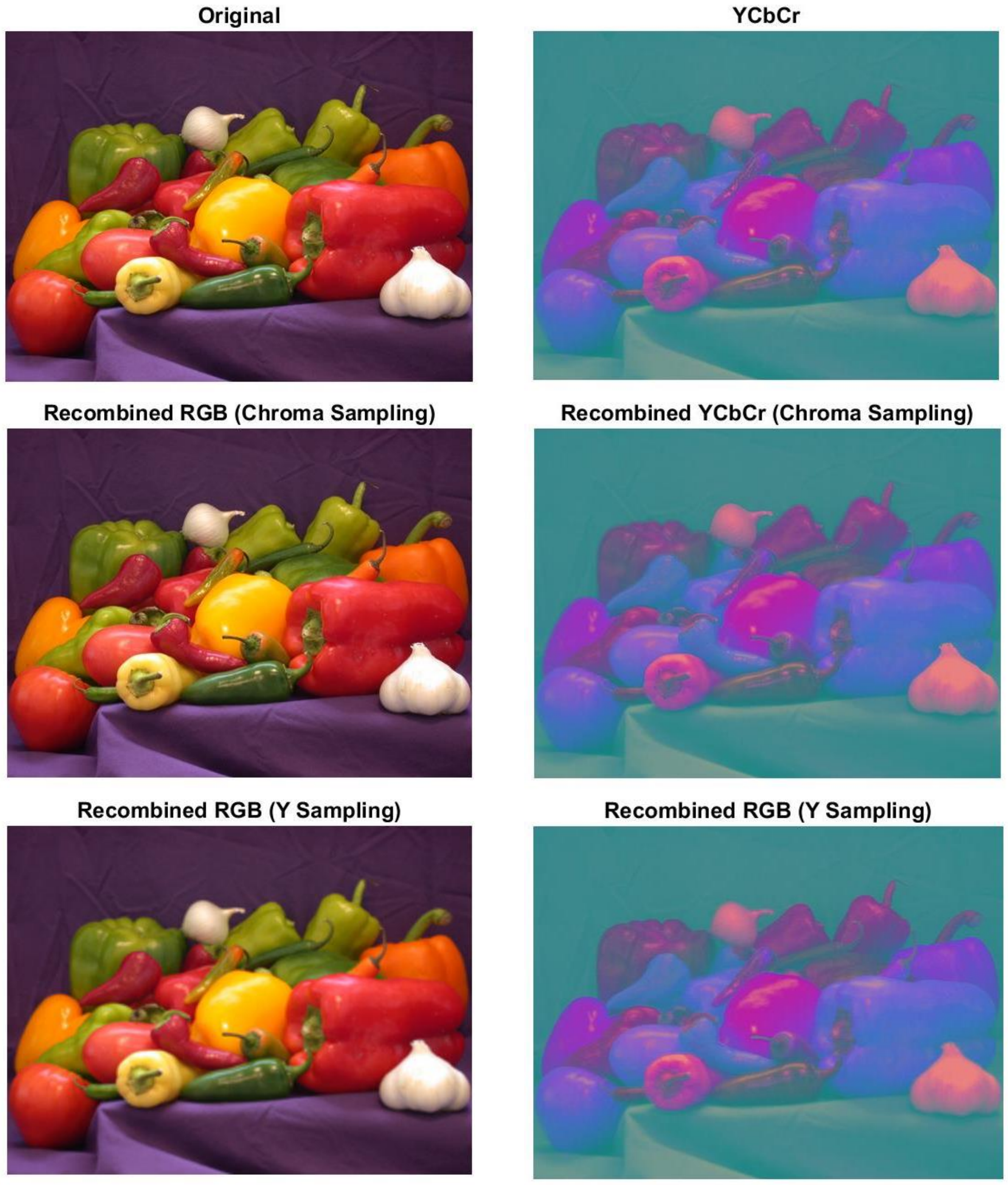

Original
YCbCr
Recombined RGB (Chroma Sampling)
Recombined YCbCr (Chroma Sampling)
Recombined RGB (Y Sampling)
Recombined RGB (Y Sampling)

Figure 2-4 Comparison of original, chroma subsampled, and Y subsampled images

3. The difference between the chroma sub-sampled images and the originals is essentially indistinguishable, as observed in Figure 2-4. There is no noticeable loss of detail or sharpness and the colour looks the same as the original RGB and YCbCr images.

4. Based on the resulting images, chroma sub-sampling has a negligible effect on image quality. This is because the human visual system is less sensitive to changes in colour; therefore, any differences that resulted from down-sampling and up-sampling the chroma channels is visually imperceptible.

5. The luma sub-sampled image (Figure 2-4) looks noticeably out of focus compared to the original image. The sub-sampled image has lost detail and edge sharpness, which has degraded the image quality.

6. Based on the resulting image, luma sub-sampling has a negative effect on image quality. Luma sub-sampling results in an undesirable loss of image detail and edge sharpness, resulting in significantly poorer image quality. This occurs because the human visual system is sensitive to changes in intensity and is able to perceive the reduction in intensity data.

7. Comparing the results of chroma and luma sub-sampling, it is clear that chroma sub-sampling produced much better results in terms of image quality. The chroma sub-sampled images appear to be of comparable image quality as the original, while the luma sub-sampled images have degraded image quality. This proves that chroma sub-sampling is the superior method for reducing network bandwidth while preserving visual acuity. The reason for this is the sensitivity of the human visual system (HVS). The HVS is much more sensitive to changes in intensity than changes in colour. As a result, colour information can be reduced without reducing perceived image quality, where as a reduction in luminance data is far more noticeable.

3 Colour Segmentation

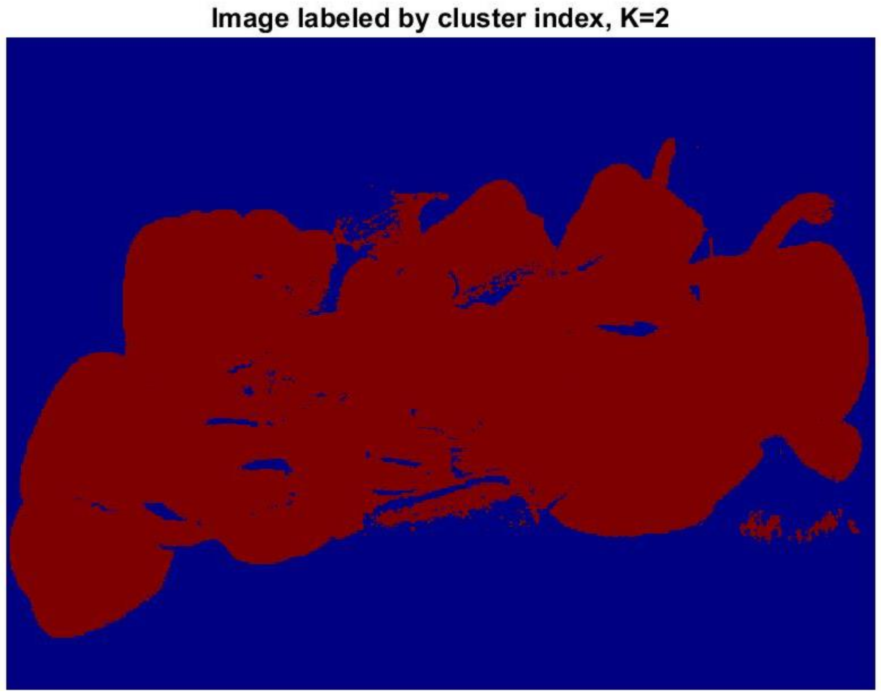


Figure 3-1 K-means clustering result with K = 2

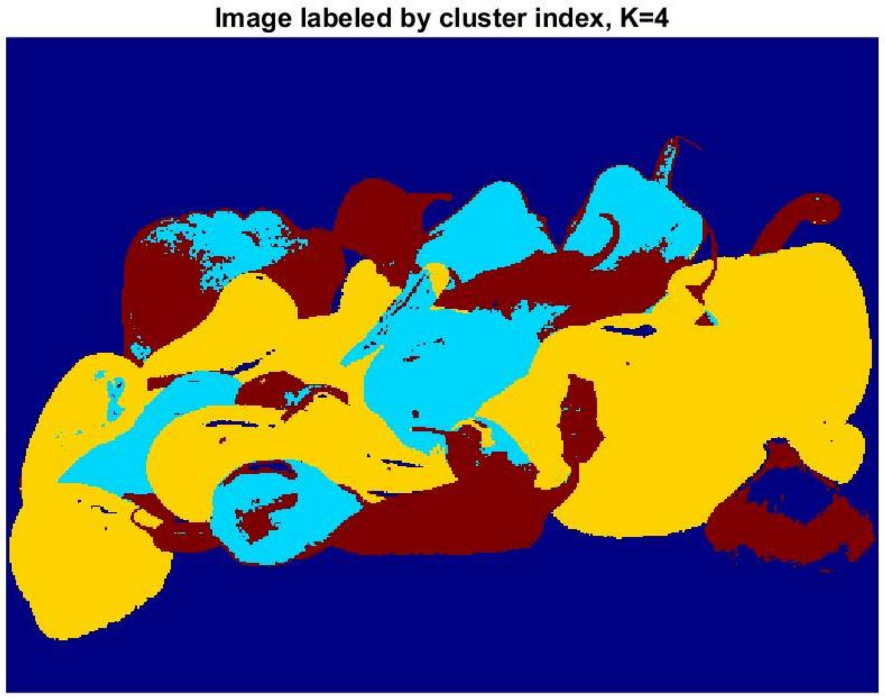


Figure 3-2 K-means clustering result with K = 4

8. The number of clusters changed with the various values of k. For $k = 2$, the resulting image has two clusters. One cluster is roughly the background, and the second cluster is roughly the peppers. For $k = 4$, the resulting image has four clusters. One cluster is roughly the background, and the three other clusters divide the peppers based on their colours, which are roughly yellow, red, and green.

   In general, the k value dictates how many clusters the image will be divided into within the relevant feature space. Clusters form around homogenous, high density areas within the feature space. In this example, the clusters form around different colours in the a* and b* colour spaces.

9. The initial points influence the final feature around which a cluster forms. If clustering is run multiple times using different initial points, the end results with vary relative to one another. This is especially true if the data is uniformly distributed without areas of obvious high density. The effect of initial points on final clusters imposes a limitation on the clustering procedure because final clusters may result from local, rather than global minima.

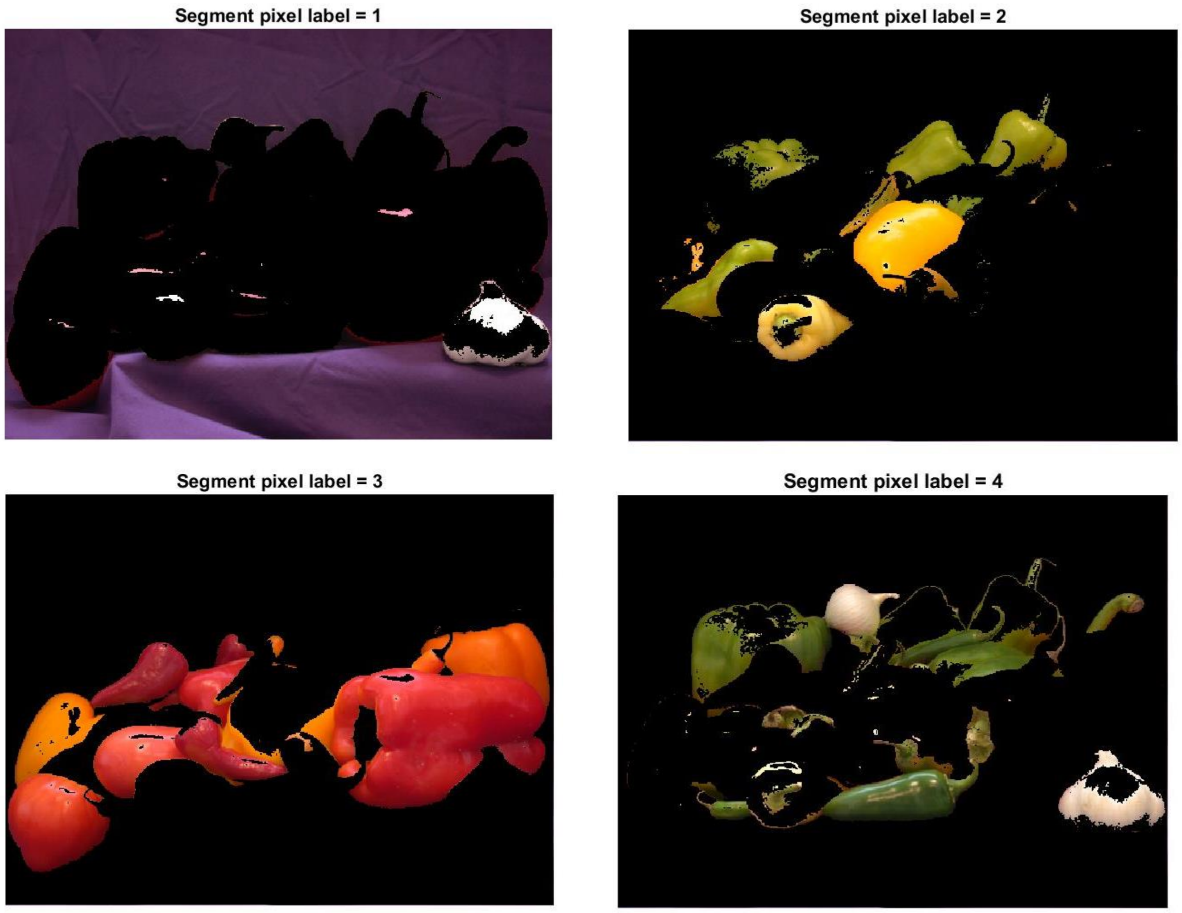


Figure 3-3 Segmented regions with original colours

10. Figure 3-3 shows the clustered regions resulting from $k = 4$ using the original colours of the peppers image. The clusters are approximately grouped around the colours purple, yellow, red, and green.

    The first cluster primarily includes the purple background drapery as well as part of the white garlic bulb and highlights on the peppers. The segmentation was fairly effective at separating the edible subjects from the purple background, with the exception of the garlic bulb.

    The second cluster isolates the yellow pepper as well as parts of other peppers that are lime green/chartreuse. The third cluster isolates the red and orange peppers. Finally, the fourth pepper isolates the green peppers and parts of the garlic bulbs.

    The clustering was fairly effective at segmenting the image by colours; however, the segmentation is less effective in certain areas. In particular, the segmentation is less clear between green, yellow, and white values. There are also pixel gaps within some of the segments. The segments could be cleaned up using morphological operations.

4 Image Transform

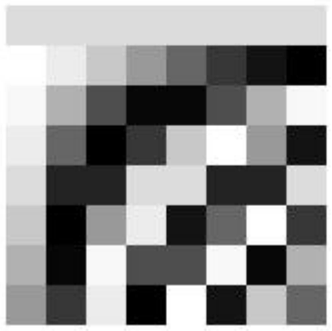


Figure 4-1 8x8 DCT transformation matrix

11. Each row of the DCT transformation matrix represents a cosine wave. The frequency of the cosine wave increases with each row, as shown in Figure 6.13a) below from the textbook, Digital Image Processing 4th Edition by Gonzalez and Woods.

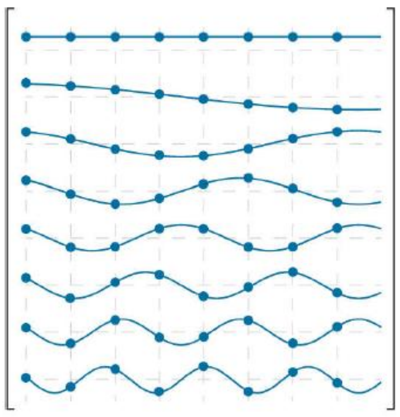


Figure 4-2 DCT transformation matrix representation (Fig. 6.13a from the textbook)

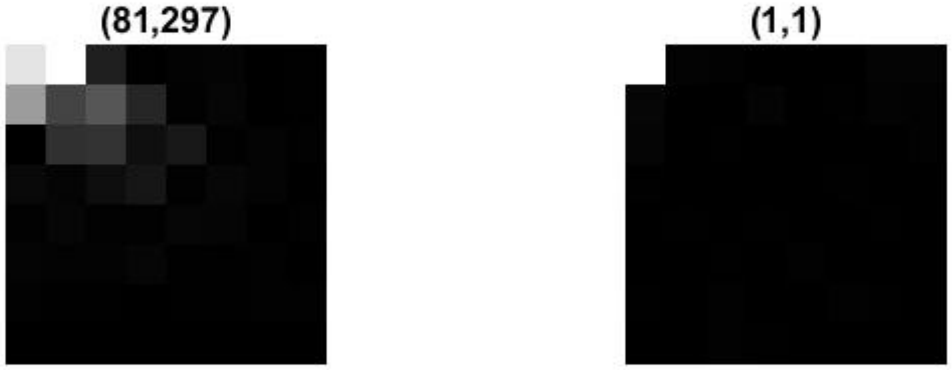


Figure 4-3 Absolute value of the DCT of sub-images at two different locations

12. The energy in the DCT sub-images is concentrated in the top left corner. The pixel values in each sub-image are the DCT coefficient values. DCT coefficients represent the weighted contribution of their corresponding element of the DCT transform matrix. The top left corner pixel is referred to as the DC coefficient and represents the DC gain of the sub-image.

    The DCT would be useful for image compression because the frequencies that contribute most to each sub-image can be identified and maintained, while the frequencies that contribute less can be discarded. This reduces the amount of information that must be transmitted while preserving the most important information needed to reconstruct the image post-transmission.

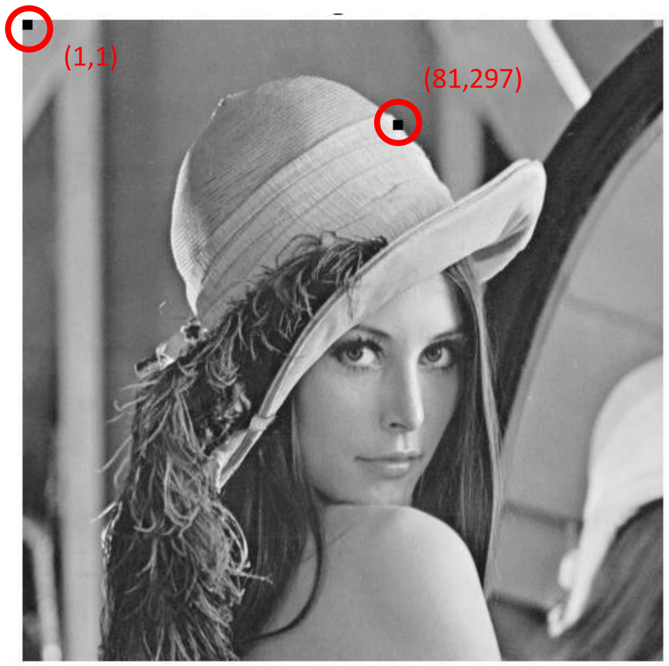


Figure 4-4 Locations of sub-images

13. The DCT of the two sub-images both show energy concentrated in the top left corner. The DCT at (1,1) shows energy only at the DC coefficient, while the sub-image at (81,297) shows energy in additional frequencies. This matches the characteristics of the of the original image at the sub-image locations (Figure 4-4). The original sub-image at (1,1) is a flat area of uniform colour, characterized by DC gain. The original sub-image at (81,297) is located near an edge, characterized by higher frequencies in addition to DC gain.

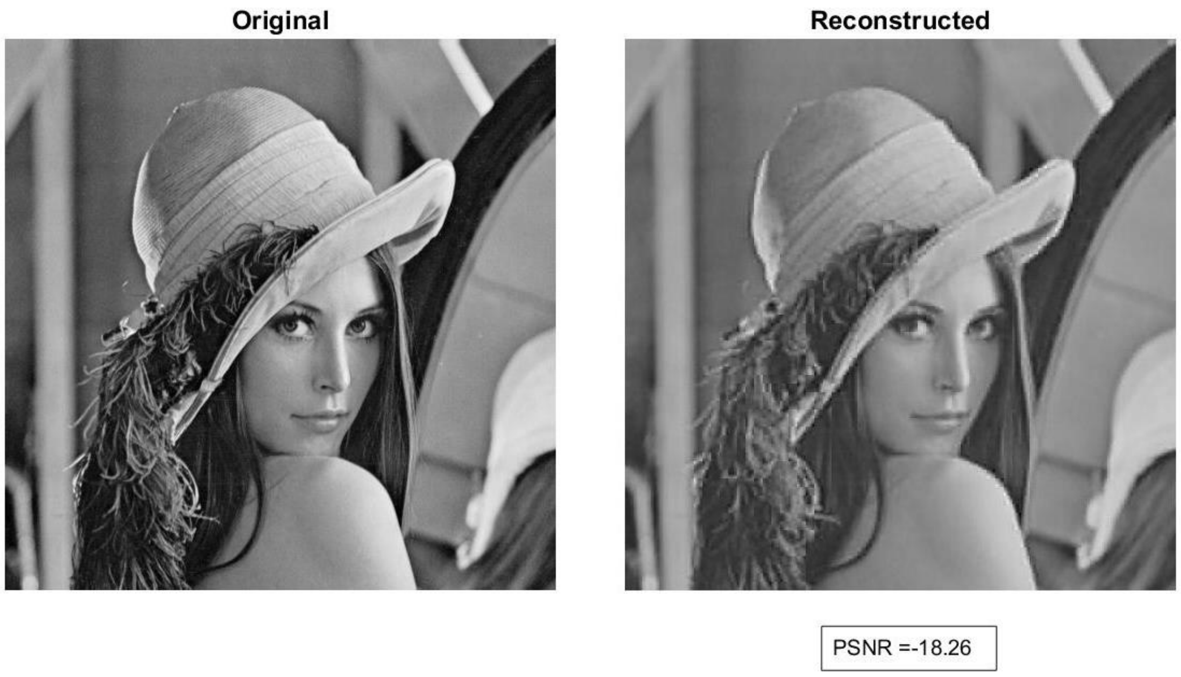


Figure 4-5 Comparison of original and reconstructed images

14. Compared to the original image, the reconstructed image looks less sharp and more muted in intensity. The reason for this is that high frequency information was discarded, and intensity was averaged out within each sub-image. High frequency information contributed to details and edges; therefore, discarding high frequency information reduced image detail and edge sharpness. This is reflected in the negative PSNR value, indicating that the image quality of the reconstructed is lower than the image quality of the original image.

15. The artifact most prominent in the image is the blocking artifact, which occurs when the boundaries between sub-images are visible. When frequency data is discarded from the sub-images, the resulting sub-image pixel values may be discontinuous with neighbouring blocks. The boundary between the blocks becomes visible, giving the image a checkered/blocky appearance.

16. The DCT is a very effective and efficient method of image compression in terms of both information packing and computational complexity. The DCT identifies the frequencies that contribute most to an image, which enables insignificant frequencies to be discarded. This reduces the amount of information needed to transmit and reconstruct the image. The amount of information contained in the transmitted image can be modified as needed by changing the number of DCT coefficients maintained in the sub-images.

5 Quantization

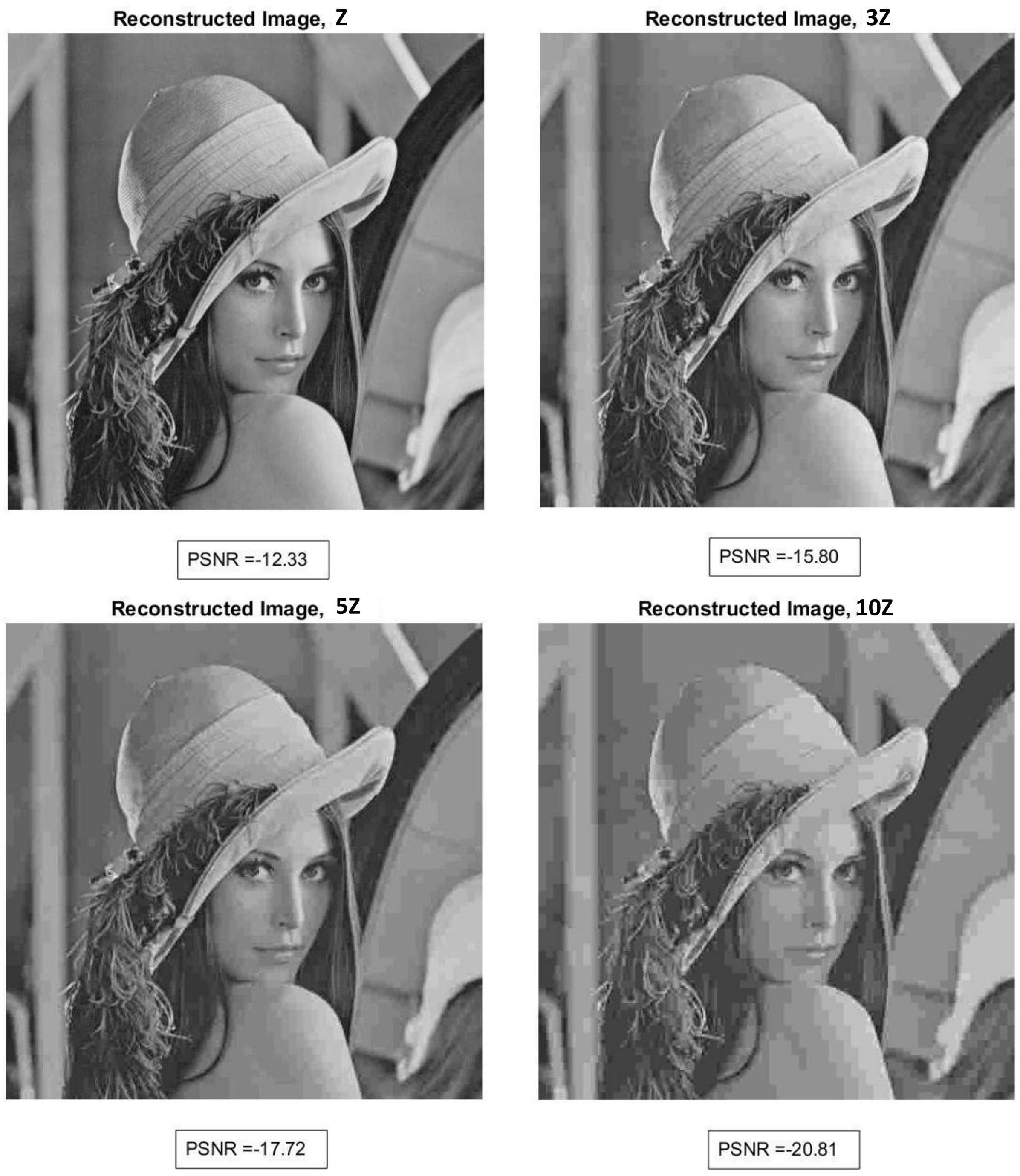


Figure 5-1 Quantization results

17. When quantization is performed, each 8x8 sub-image is divided by the quantization matrix in an element-wise fashion. Next, the resulting values are rounded to the nearest integer. The quantization matrix is constructed in accordance with the human visual system, which is more sensitive to lower frequency image data than higher frequency. Quantization matrix values are chosen such that lower frequency DCT coefficients are divided by smaller

numbers and higher frequency coefficients are divided by larger values. When rounding occurs, the often reduces higher frequency coefficients to zero. This means that lower frequencies are given more weight in the image while higher frequencies are often discarded. As a result, quantized images have lower image qualities compared to their original, and negative PSNR values as shown in Figure 5-1.

18. In the reconstructed image using 3Z, blocking artifacts are becoming more noticeable. For example, the smooth, continuous background in the original has blocks of different shades of grey in the reconstructed image. This is because the multiple on Z has increased the amount of quantization occurring. More high frequency information is truncated leading to more discontinuities among neighbouring sub-images.

19. As the multiple of Z increases, so does the level of quantization and discarded high frequency data. As shown in Figure 5-1, as quantization increases, the level of blocking in the image increases. This produces a posterizing effect most visible for 10Z in Figure 5-1, in which continuous colour gradients in the original image become regions of fewer tones with abrupt changes at their border. This is also evident in the PSNR values, which also become poorer (i.e., more negative) as the quantization multiple increases.

20. As the level of quantization increases, blocking artifacts become more prominent. Higher levels of quantization remove more mid and high frequencies from the sub-image, which removes detail and makes the intensity of the block pixels more uniform. The effect of quantization may have different results in neighbouring sub-images, causing a visible discontinuity at their border. At high quantization levels, the reconstructed image can appear posterized as the subtle variations in the original pixels become distinct, uniform regions.

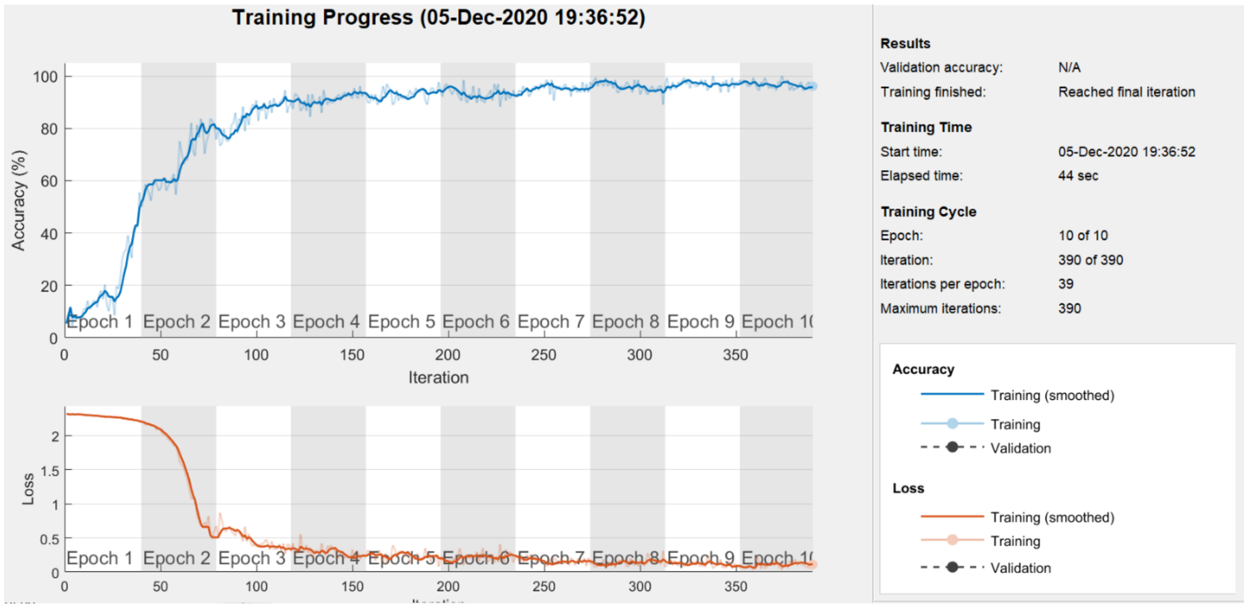
21. Based on these results, it is evident that there is a trade-off between compression performance and image quality. As the level of quantization increases, image compression increases. However, increased image compression results in poorer image quality. With increased quantization and compression, image detail is lost and intensity becomes more muted.

6 Convolutional Neural Networks

22. In general, a fully connected layer is usually more expensive than a convolutional layer. For fully connected layers, every neuron in one layer is connected to every neuron in the previous/next layer. This means that the number of parameters increases significantly with the size of the input image. A high number of parameters also results in an increased runtime compared to convolutional layers because there are more parameters to be tuned.

    A convolutional layer is beneficial for images because it takes advantage of the spatial correlation of pixels. A convolutional layer processes smaller patches of the image via convolution (or correlation) and does not require as many parameters as fully connected.

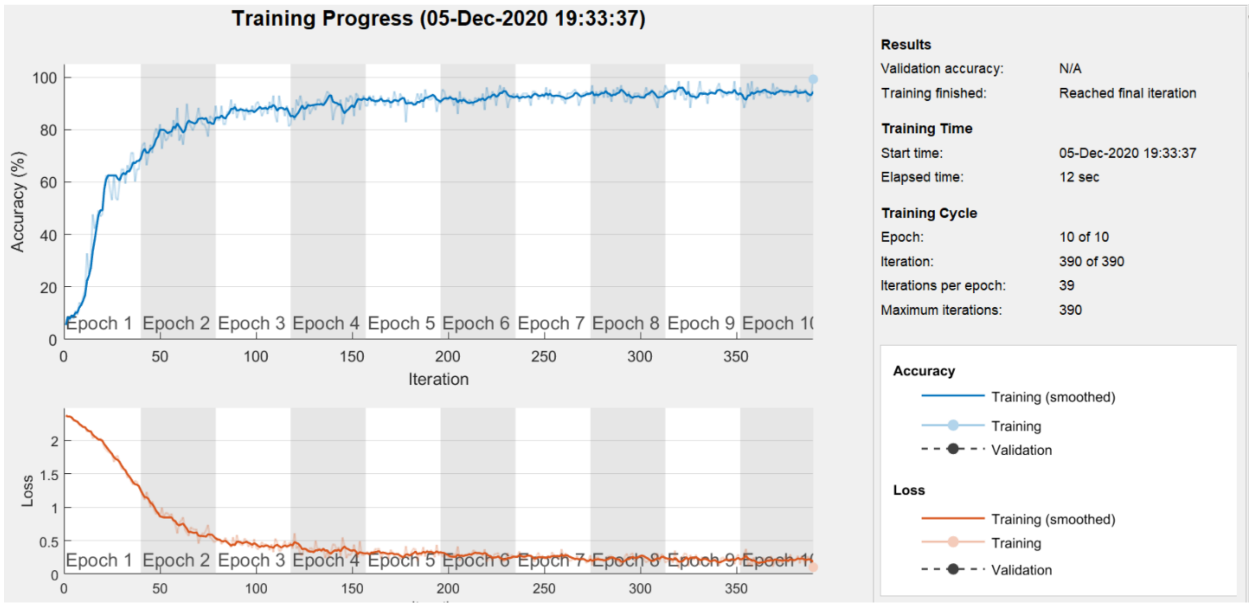
Note: When running the provided CNN and MLP code, the CNN was observed to have a much longer runtime than the MLP. The runtime of a complete network is a function of both the number and type of layers, and a given network can include both fully connected and convolutional layers among others. This is the case for the CNN.


Training Progress (05-Dec-2020 19:36:52)
Accuracy (%)
Loss
Iteration
Epoch 1 Epoch 2 Epoch 3 Epoch 4 Epoch 5 Epoch 6 Epoch 7 Epoch 8 Epoch 9 Epoch 10
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

Results

Validation accuracy:	N/A
Training finished:	Reached final iteration
Training Time	
Start time:	05-Dec-2020 19:36:52
Elapsed time:	44 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	390 of 390
Iterations per epoch:	39
Maximum iterations:	390

Figure 6-1 CNN results


Training Progress (05-Dec-2020 19:33:37)
Accuracy (%)
Loss
Iteration
Epoch 1 Epoch 2 Epoch 3 Epoch 4 Epoch 5 Epoch 6 Epoch 7 Epoch 8 Epoch 9 Epoch 10
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

Results

Validation accuracy:	N/A
Training finished:	Reached final iteration
Training Time	
Start time:	05-Dec-2020 19:33:37
Elapsed time:	12 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	390 of 390
Iterations per epoch:	39
Maximum iterations:	390

Figure 6-2 MLP results

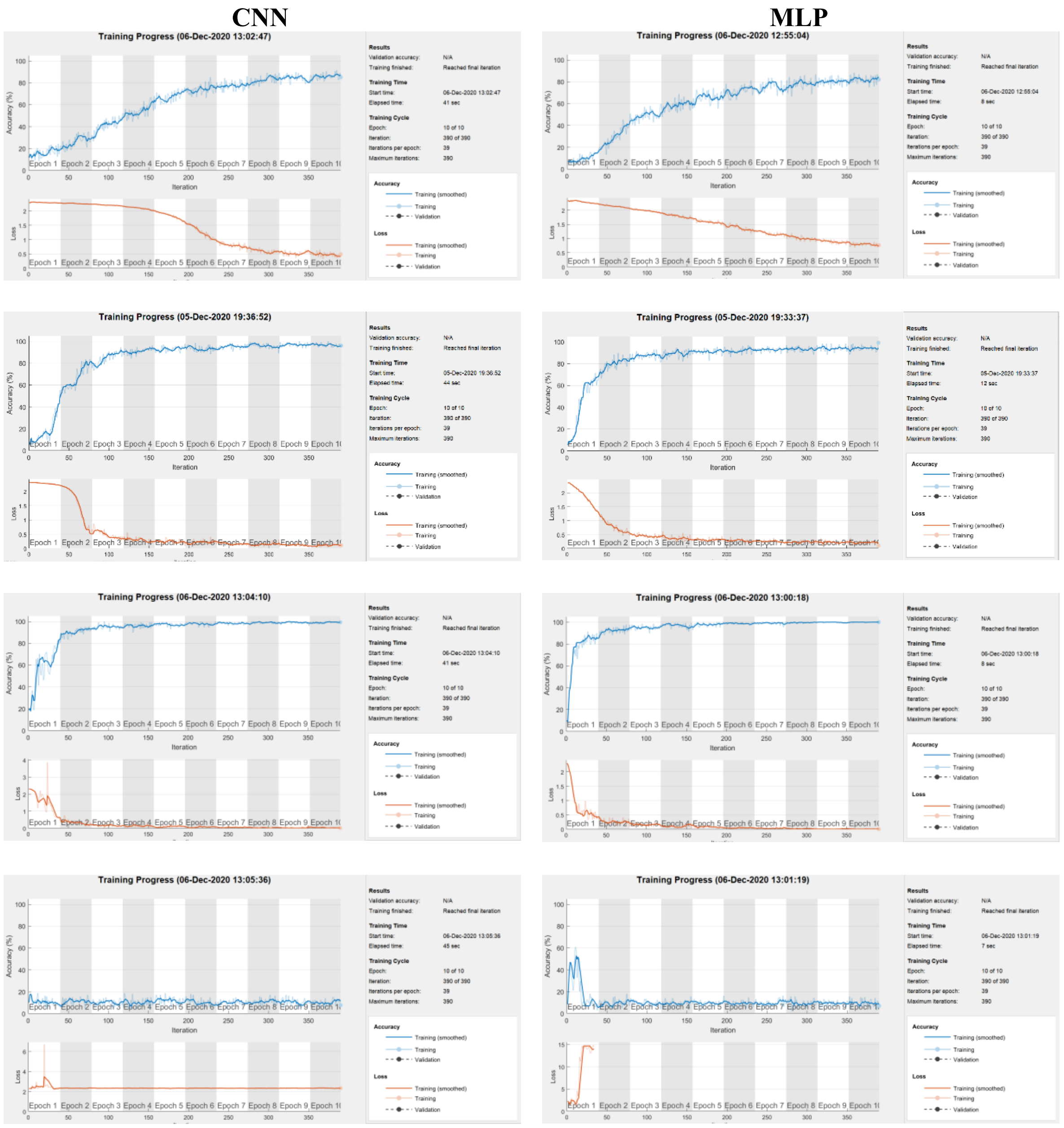
Table 6-1CNN and MLP accuracy

	CNN	MLP
Training accuracy	0.9726	0.9464
Testing accuracy	0.9509	0.9049

23. Both the CNN and MLP achieve most of their training within two epochs. The CNN training rate was initially slow, and then rapidly increased during the second epoch. The MLP learned at a more constant rate during the first two epochs. The overall training time duration was longer for the CNN at 44 seconds versus the MLP at 12 seconds.

    With a learning rate of 0.01 and 10 epochs, both the CNN and MLP converged. The CNN performed better, converging to a higher training accuracy of 0.9726 compared to 0.9464 for the MLP. The CNN also performed more accurately during testing at 0.9509 compared to 0.9049 for MLP.

24. The testing accuracy was similar to, but lower than the training accuracy for both the CNN and MLP. For the CNN, the difference was roughly 0.02. For the MLP, the difference was double that of the CNN at roughly 0.04.


CNN
MLP
Training Progress (06-Dec-2020 13:02:47)
Training Progress (06-Dec-2020 12:55:04)
Training Progress (05-Dec-2020 19:36:52)
Training Progress (05-Dec-2020 19:33:37)
Training Progress (06-Dec-2020 13:04:10)
Training Progress (06-Dec-2020 13:00:18)
Training Progress (06-Dec-2020 13:05:36)
Training Progress (06-Dec-2020 13:01:19)
Accuracy (%)
Loss
Iteration
Epoch 1 Epoch 2 Epoch 3 Epoch 4 Epoch 5 Epoch 6 Epoch 7 Epoch 8 Epoch 9 Epoch 10
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 13:02:47
Elapsed time:             41 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 12:55:04
Elapsed time:             8 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               05-Dec-2020 19:36:52
Elapsed time:             44 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               05-Dec-2020 19:33:37
Elapsed time:             12 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 13:04:10
Elapsed time:             41 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 13:00:18
Elapsed time:             8 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 13:05:36
Elapsed time:             45 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

```
Results
Validation accuracy:      N/A
Training finished:        Reached final iteration
Training Time
Start time:               06-Dec-2020 13:01:19
Elapsed time:             7 sec
Training Cycle
Epoch:                    10 of 10
Iteration:                390 of 390
Iterations per epoch:     39
Maximum iterations:       390
```

Figure 6-3 Comparison of initial learning rates of 0.001 (row 1), 0.01 (row 2), and 0.1 (row 3) and 1.0 (row 4)

25. Figure 6-3 shows the results of running the CNN and MLP with training rates raised and lowered by factors of 10. For both the CNN and MLP, the time to converge increased as the initial learning rate was decreased. It is evident that as the learning rate decreases, the number of epochs needed to achieve convergence increases.

The training graphs were stable and converged for initial learning rates of 0.001, 0.01, and 0.1 but did not converge for an initial learning rate of 1.0. For 1.0, the training graphs did not show any improvement or evidence of learning. It seems that this learning rate was too high for the model to be trained.

Table 6-2 Accuracy results for 10 and 100 epochs

	CNN		MLP	
Epochs	10	100	10	100
Training accuracy	0.9726	1	0.9464	1
Testing accuracy	0.9509	0.9729	0.9049	0.9322
Difference	0.0219	0.0271	00415	0.0678

26. Overfitting can be identified when the neural network produces high training accuracy but very different testing accuracy. Due to the higher number of parameters, the MLP is more susceptible to overfitting than the CNN.

    Table 6-2 compares the accuracy results for the CNN and MLP trained with 10 and 100 epochs. The difference between the training and testing accuracy for the CNN is in the range of 0.02 to 0.03. For the MLP, the difference is greater, at approximately 0.04 for 10 epochs and 0.07 for 100 epochs. Some error is expected; however, overfitting may be occurring, especially for the MLP with 100 epochs since the training accuracy is 100% but the accuracy on new data is 7% lower. This suggests that due to the greater number of epochs, the model parameters adapted to fit the biases and noise in the training data too closely. Consequently, new data, with different biases and noise, results in poorer performance.

7 Conclusions

This lab provided hands-on experience with image compression, colour segmentation, neural networks, and practice using image processing tools in MATLAB.

From this lab, it is evident that chroma sub-sampling is the most effective method for reducing the amount of data in an image without degrading image quality. This is because the human visual system is less sensitive to changes in colour than changes in intensity.

Colour segmentation groups pixels within the three-dimensional RGB colour space. In k-means clustering, the number of segments is dictated by the value of k. This method is limited in that the results vary based on the initial positions of the cluster means.

The DCT is an effective image transform in the image compression process. It enables the identification of the most important frequencies in an image, which are usually low frequencies. This makes it possible to discard unnecessary information through a process like quantization. The result is a much smaller image in terms of information packing. The downside of image compression is blocking artifacts, which become more prominent as the level of quantization increases.

Finally, convolution neural networks are beneficial for processing images because they take advantage of the spatial correlation of pixels, resulting in fewer parameters, and less risk of overfitting. The rate of convergence and level of accuracy is a function of the initial learning rate and number of epochs. A faster learning rate requires fewer epochs to train the model, but if the learning rate is too high, convergence will not occur.

Appendix – MATLAB Code

File: lab5_prt2_chroma_subsampling.m

```
clc; clear; close all;
fontSize = 12;

% Load peppers image, convert to YCbCr, and plot both
RGB = imread('peppers.png');
YCbCr = rgb2ycbcr(RGB);
figure(1); imshow(RGB); title('Original', 'FontSize', fontSize);
figure(2); imshow(YCbCr); title('YCbCr', 'FontSize', fontSize);

% Plot Y, Cb, and Cr channels separately
Y = YCbCr(:,:,1);
Cb = YCbCr(:,:,2);
Cr = YCbCr(:,:,3);
figure(3); imshow(Y); title('Y Channel', 'FontSize', fontSize);
figure(4); imshow(Cb); title('Cb Channel', 'FontSize', fontSize);
figure(5); imshow(Cr); title('Cr Channel', 'FontSize', fontSize);

% Reduce resolution of chroma channels by factor of 2
Cb_down = imresize(Cb, 1/2);
Cr_down = imresize(Cr, 1/2);

% Upsample chroma channels to original resolution
Cb_up = imresize(Cb_down, 2, 'bilinear');
Cr_up = imresize(Cr_down, 2, 'bilinear');

% Recombine Y, Cb, and Cr channels
YCbCr_recombined1 = cat(3, Y, Cb_up, Cr_up);
RGB_recombined1 = ycbcr2rgb(YCbCr_recombined1);
figure(6); imshow(YCbCr_recombined1);
title('Recombined YCbCr (Chroma Sampling)','FontSize', fontSize);
figure(7); imshow(RGB_recombined1);
title('Recombined RGB (Chroma Sampling)','FontSize', fontSize);

% Reduce resolution of Y channel by factor of 2
Y_down = imresize(Y, 1/2);

% Upsample Y channel to original resolution
Y_up = imresize(Y_down, 2, 'bilinear');

% Recombine Y, Cb, and Cr channels
YCbCr_recombined2 = cat(3, Y_up, Cb, Cr);
RGB_recombined2 = ycbcr2rgb(YCbCr_recombined2);
figure(8); imshow(YCbCr_recombined2);
title('Recombined YCbCr (Y Sampling)','FontSize',fontSize);
figure(9); imshow(RGB_recombined2);
title('Recombined RGB (Y Sampling)','FontSize',fontSize);
```

File: lab5_prt3_colour_segmentation.m

```
clc; clear; close all;
fontSize = 12;

% Load peppers image, convert to the L*a*b* colour space
f = imread('peppers.png');
size(f)
C = makecform('srgb2lab');
im_lab = applycform(f,C);

% Reshape the a* and b* channels
ab = double(im_lab(:,:,2:3));
m = size(ab,1);
n = size(ab,2);
ab = reshape(ab,m*n,2);

% Classify the colour using k-means clustering
%K = 2;
%row = [55 200];
%col = [155 400];

K = 4;
row = [55 130 200 280];
col = [155 110 400 470];

% Convert (r,c) indexing to 1D linear indexing
idx = sub2ind([size(f,1) size(f,2)], row, col);

% Vectorize starting coordinates
for k = 1:K
    mu(k,:) = ab(idx(k),:);
end

cluster_idx = kmeans(ab,K,'Start',mu);

% Label each pixel according to k-means
pixel_labels = reshape(cluster_idx, m, n);
h = figure; imshow(pixel_labels, []);
title(['Image labeled by cluster index, K=',num2str(K)],'FontSize',fontSize);
colormap('jet');

% Output each segment with original image pixels
segment = f;

% Change label value (1,2,3, or 4) to show different segment
label = 4;

for i = 1:m-1
    for j = 1:n-1
        if(pixel_labels(i,j) ~= label)
            segment(i,j,1) = 0;
            segment(i,j,2) = 0;
            segment(i,j,3) = 0;
        end
    end
end

figure; imshow(segment, []);
title(['Segment pixel label = ', num2str(label)]);
```

File: lab5_prt4_image_transform.m

```
clc; clear; close all;
fontSize = 12;

% Load Lena image
lena = imread('lena3.tiff');
f = double(rgb2gray(lena));

% Construct and plot DCT transform matrix
T = dctmtx(8);
figure(1); imshow(imresize(T,20,'nearest'), []);

% Apply the DCT transformation
F_trans = floor(blkproc(f-128,[8 8],'P1*x*P2',T,T'));
sub1 = F_trans(81:81+7,297:297+7);
sub2 = F_trans(1:8,1:8);
figure(2); imshow(imresize(abs(sub1),20,'nearest'),[]); title('(81,297)');
figure(3); imshow(imresize(abs(sub2),20,'nearest'),[]); title('(1,1)');

% Discard all but 6 of the DCT coefficients and reconstruct image
mask = zeros(8,8);
mask(1,1) = 1;
mask(1,2) = 1;
mask(1,3) = 1;
mask(2,1) = 1;
mask(3,1) = 1;
mask(2,2) = 1;
F_thresh = blkproc(F_trans, [8 8], 'P1.*x', mask);
f_thresh = floor(blkproc(F_thresh, [8 8], 'P1*x*P2', T', T)) + 128;

% Plot original and reconstructed images and PSNR
figure(4); imshow(f,[]); title('Original','FontSize', fontSize);
figure(5); imshow(f_thresh,[]); title('Reconstructed','FontSize', fontSize);
psnr = PSNR_norm(f,f_thresh);
txt = strcat('PSNR = ', sprintf('%.2f',psnr));
annotation('textbox', [0.4, 0, 0.1, 0.1], 'string', txt);
```

File: lab5_prt5_quantization.m

```
clc; clear; close all;
fontSize = 12;

% Load Lena image
lena = imread('lena3.tiff');
f = double(rgb2gray(lena));

% Construct quantization matrix (change multiple to 1, 3, 5, 10)
multiple = 1;
Z = multiple.*[16 11 10 16 24 40 51 61;
               12 12 14 19 26 58 60 55;
               14 13 16 24 40 57 69 56;
               14 17 22 29 51 87 80 62;
               18 22 37 56 68 109 103 77;
               24 35 55 64 81 104 113 92;
               49 64 78 87 103 121 120 101;
               72 92 95 98 112 100 103 99];

% Construct and plot DCT transform matrix
T = dctmtx(8);

% Apply the DCT transformation
F_trans = floor(blkproc(f-128,[8 8],'P1*x*P2',T,T'));
```

```
% Perform quantization and calculate PSNR
fun1 = @(block_struct) round(double(block_struct) ./ Z);
F_quant = blkproc(F_trans,[8 8],fun1);

%Reconstruct the image and calculate PSNR
fun2 = @(block_struct) T'*(block_struct .* Z)*T;
f_recon = blkproc(F_quant, [8 8], fun2) + 128;
psnr = PSNR_norm(f,f_recon);

% Plot reconstructed image
figure; imshow(f_recon,[]);
title(['Reconstructed Image, ', num2str(multiple), 'Z']);
txt = strcat('PSNR = ', sprintf('%.2f',psnr));
annotation('textbox', [0.4, 0, 0.1, 0.1], 'string', txt);
```

File: PSNR_norm.m

```
function [PSNR_out] = PSNR_norm(f,g)
    PSNR_out = 10*log10(1/mean2((f-g).^2));
end
```

File: CNN.m (provided)

```
clc
clearvars

load mnist_.mat
Xtrain = double(mnist.train_images)/255.0-0.5;
Xtrain = reshape(Xtrain,28,28,1,5000);
ytrain = categorical(mnist.train_labels);
Xtest = double(mnist.test_images)/255.0-0.5;
Xtest = reshape(Xtest,28,28,1,10000);
ytest = categorical(mnist.test_labels);

for i = 1:25
    subplot(5,5,i);
    imshow(Xtrain(:,:,i),[]);
end

% Define Network Architecture
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')

    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

```
cell_test = {Xtest ytest};
% Specify Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',10, ...
    'Shuffle','every-epoch', ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');

% Train Network Using Training Data
net = trainNetwork(Xtrain, ytrain, layers, options);

% Classify Validation Images and Compute Accuracy
YPred = classify(net,Xtest);
'Testing accuracy'
test_accuracy = sum(YPred == ytest)/numel(ytest)

'Training accuracy'
YPred = classify(net, Xtrain);
train_accuracy = sum(YPred == ytrain)/numel(ytrain)
```

File: mlp.m (provided)

```
clc
clearvars

load mnist_.mat
Xtrain = double(mnist.train_images)/255.0 - 0.5;
Xtrain = reshape(Xtrain,28,28,1,5000);
ytrain = categorical(mnist.train_labels);
Xtest = double(mnist.test_images)/255.0 - 0.5;
Xtest = reshape(Xtest,28,28,1,10000);
ytest = categorical(mnist.test_labels);

for i = 1:25
    subplot(5,5,i);
    imshow(Xtrain(:,:,i),[]);
end

% Define Network Architecture
layers = [
    imageInputLayer([28 28 1])

    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(10)

    softmaxLayer
    classificationLayer];
cell_test = {Xtest ytest};
% Specify Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',10, ...
    'Shuffle','every-epoch', ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

```
% Train Network Using Training Data
net = trainNetwork(Xtrain, ytrain, layers, options);

% Classify Validation Images and Compute Accuracy
YPred = classify(net,Xtest);
'Testing accuracy'
test_accuracy = sum(YPred == ytest)/numel(ytest)

'Training accuracy'
YPred = classify(net, Xtrain);
train_accuracy = sum(YPred == ytrain)/numel(ytrain)
```