SYDE 575

Group #22

Lab 5 Report

Prof David A. Clausi

Abdulhameed Naji - *20642623*

Shilpan Shah - *20603389*

Contents

List of Figures

1 Introduction

The following lab investigates methods and consequences of image compression such as the ones that are part of the JPEG pipeline. Images were transformed to YCbCr space to separate the structural info (Y-channel) and color info (CbCr). Aggressive compression is then applied to the color channels to compress the image with little to no observable losses.

K-mean clustering is used to segment images based on their relative color. The classification of an arbitrary number of cluster was based on distinct regions formed by the histogram of the image. The K-mean method was effective at separating foreground and background content.

In an effort to understand the JPEG compression pipeline, the DCT transform is performed on a sample image and then quantized. The DCT transform embeds the cosine components that form the 8x8 image patch. Quantization then attenuates higher frequency term such that they can be neglected in the image representation. This in turn increases the compression at the cost of blocking artifacts due to less fine detail representation. The artifacts were directly proportional to the intensity of quantization.

Finally, Convolutional Neural Networks were explored in conjunction with Multi-Layer Perceptrons (MLP). Both models are trained and test against a data set. MLP offered higher complexity but faster training times. On the other hand, the lesser parameters used in CNNs reduced the presence of over-fitting in large training epochs.

2 Chroma Subsampling

2.1 Q1

The separated channels are shown in Figure 1.

Figure 1: Extracted YCbCr channels

By definition, the Cb channel is the blue-difference whereas the Cr channel is the red-difference. They describe the relative representation of blue and red components in the image with respect to the luma. For example, notice the Cr channel shows brighter readings for the peppers colored red. This is because the channel's output is represented by:

$$Cr = R - Y$$

Subtracting the luma component gives the relative intensity of the chroma component; hence, for a red saturated object, the corresponding luma is dark. This yields a Cr output that is bright.

Conversely, the yellow pepper has a high/bright luma component. The color blue is not present in a yellow color as it is the intersection of red and green. As a result, the output of the Cb channel for the yellow pepper is small.

2.2 Q2

The Y-Channel is essentially a grey scale version of the image, and hence contains all the necessary fine details to describe the structure of the image. The Cb and Cr components exist to describe the color information in the image - their sole purpose is to reduce the redundancy in the RGB representation by embedding the color information of the image in just two component.

2.3 Q3

Figure 2 shows the result of compressing the CbCr channels. The images appear virtually unchanged regardless of the lossy down sampling operation. This makes sense as we formerly stated that the channels are responsible for color information whereas the luma (Y) is responsible for structure. Since the human visual system is less sensitive to changes in color, we do not pick up on changes or losses in color information from this compression.

Figure 2: Resulting image with compressed CbCr channels

2.4 Q4

Based on the explanation above, chroma sub-sampling has little to no effect on image integrity as the structural detail is left unaffected. Color information is aggressively compressed, but the difference is not perceived by the human visual system.

2.5 Q5, Q6

Figure 3: Resulting image with compressed Y channels

Figure 3 shows the result of compressing the Y channel. Since the Y channel is responsible for the structural information, the down sampling procedure produced a blur that is visible in the resulting compressed image. The human visual system is sensitive to edge detail being lost. The difference is very clear around the garlic bulb. Notice the blurred detail as seen in Figure 4.

Figure 4: Edge comparison between the compressed (left) and original (right) image feature.

2.6 Q7

Chroma sub-sampling is superior when compared to luma sub-sampling. Since we are insensitive to color reduction, the compression can be carried out without any notable artifacts or impact on integrity. Moreover, it is far more efficient to compress the chroma components for network bandwidth since we compress twice the amount of data. Since images scale exponentially, being able to dramatically compress two images is significantly more advantageous than compressing a single image less aggressively due to fear of losing structural information.

3 Color Segmentation

3.1 Q1

First, the image is segmented using two clusters (K=2) with results shown in Figure 5. Following that, a clustering of size K=4 is performed with results shown in Figure 6.

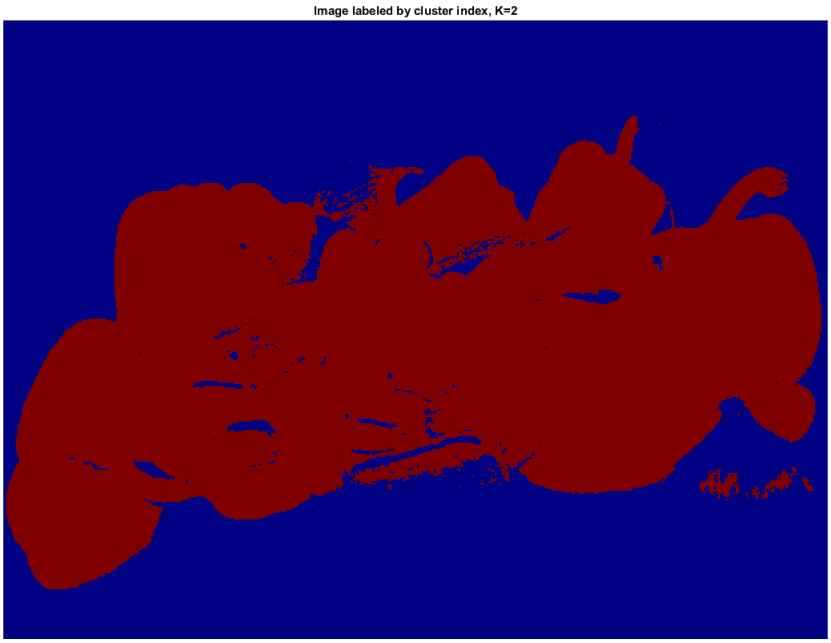

Figure 5: Depiction of clusters using K = 2

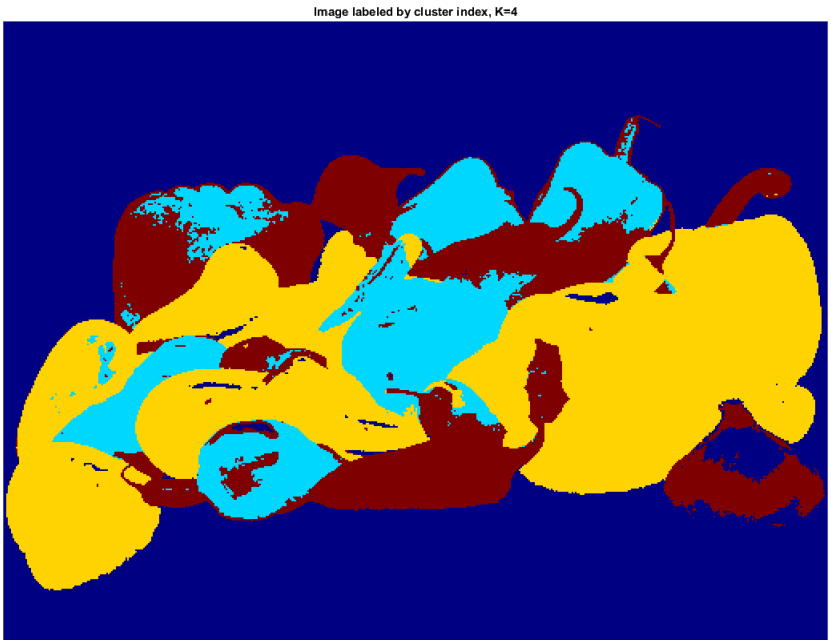

Figure 6: Depiction of clusters using K = 4

Increasing the value K implies we are classifying the image based on K number of means; hence, we expect to see K number of clusters and segmentation. In the case of K = 2, we classify pixels into two clusters whose mean best segments the image into distinct classes. Similarly, when K is increased to 4, the best means (colors) that segment the image are the four colors shown in 6. The segmentation figure shows the classification based on the original image. For instance, the red peppers were classified under the mean value formed by the yellow cluster.

3.2 Q2

For K-Means, the initial seed is usually chosen arbitrarily. This effects the initial iterations of the algorithm. However, since the algorithm is iterative, it converges to a mean value that best segments the image. As a result, the initial seed hypothetically has no effect on the final result. With that being said, the only limitation that exists is in cases where the image is fairly uniform – that is, there is no unique classification that best segments the image. In that scenario, different seed selections may produce different results since the algorithm's objective function will have no preference between classifications that produce the same level of segmentation.

3.3 Q3

The segmentation result is shown in Figures 7 and 8 for two and four clusters respectively.

Figure 7: Segmented regions using K = 2

For two clusters, the algorithm does a good job at separating the background and foreground objects. This is expected since the background is uniform and predominantly monochromatic. The background forms a locally dominant distribution in the histogram, hence, it naturally forms a mean that best segments the image. Since a cluster of size 2 does not have enough granularity to classify the relative colors of the peppers, they are binned under one classification – objects that are not the background.

Introducing a larger cluster size of 4 attempts to segment the peppers by their colors. The

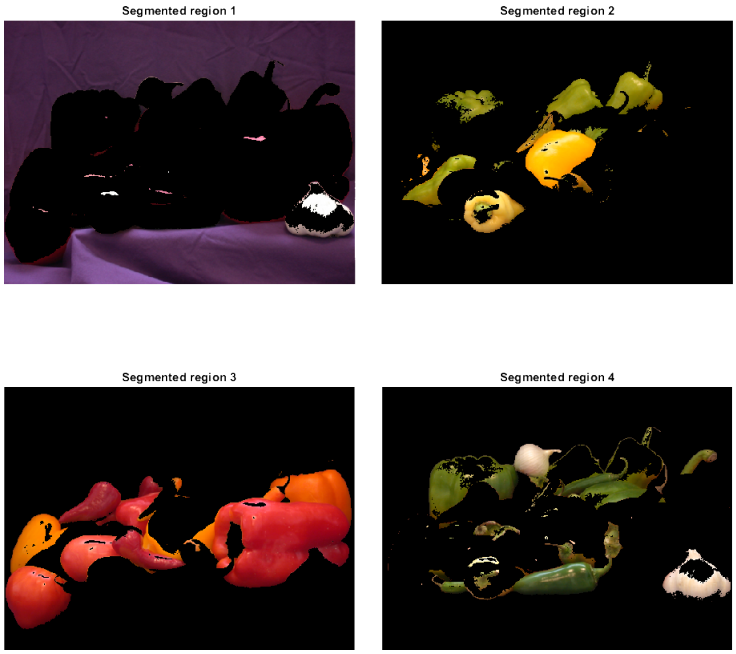

Figure 8: Segmented regions using K = 4

first notable classification is the background for the same reasons stated formerly. Another notable classification is the cluster formed by the red pepper – they form a dominant distribution in the histogram (high probability, small variance) and hence are isolated accurately. There are two remaining classes, however the remainder of the peppers have a similar hue. We can see that the classification did its best to segment the image into higher saturation and lower saturation greens. Since these colors are closely related, the segmentation is comparatively less refined compared to the red peppers. Another way to verify this is by plotting the segments for K = 3; knowing that the artifacts were generated due to the difficulty of separating closely related colors, we predict that a cluster of 3 will bin the green peppers together to reduce the overall artifact. The results are verified in Figure 9.

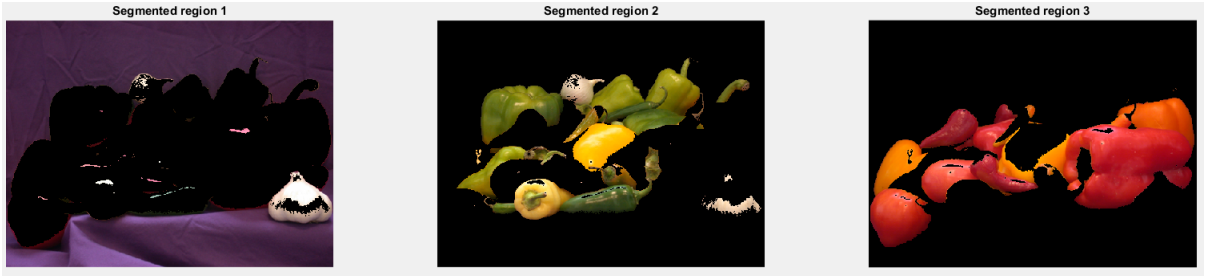

Figure 9: Segmented regions using K = 3

In effect, enforcing a classification of two similar distribution will inevitably cause overlaps between the two classes that may produce false results.

4 Image Transform

4.1 Q1

The DCT matrix is depicted in Figure 10. Going along the rows, the DCT matrix represents 1-D cosines of varying frequencies. In the first row, the pattern is solid, indicating DC gain. The second row introduces more frequencies, we see a transition high intensity to low intensity over a transition period - this corresponds to a step or ramp. As we traverse down the remaining row, the alternating patterns become more frequent; the last row has almost 1 cycle per pixel.

Figure 10: Visual depiction of DCT matrix

4.2 Q2 & Q3

The pixel (1,1) belongs to a uniform background and its transform is shown in Figure 11. Since the transform encodes the frequency of the cosine component that constitutes the im-

Figure 11: DCT of pixel at (1,1)

age patch, a uniform background is best represented by low frequencies dominated by a DC gain. This is seen in figure as the (u,v) index at (1,1) is the most dominant component - in this case, the near saturated intensity indicate that the patch of the image is best represented by a DC gain value only.

On the other hand, the region at (297, 81) is located on the edge of the hat of the lena image - the transformation is show in Figure 12. Due to the relative size of the patch, the transform primarily sees a uniform patch of intensities. However, we immediately notice a subtle introduction of higher frequency terms. The transformation at (u,v) = (1,2) and (u,v) = (2,1) have low intensity contributions. This implies that the image patch is described by a uniform color (DC gain) with very little change of intensities represented by the ramp of the higher frequency components.

Figure 12: DCT of region at (297, 81)

In both cases, we notice that the image is primarily characterized by low frequency components; a significant portion of which is the DC gain. This energy distribution of the transform is helpful because it helps identify the fact that high frequency components are unnecessary to capture the majority of structure in an image. In most cases, the high frequency components will be rounded to 0 after quantization, which will improve the compression – less bits are needed to fully describe the image.

4.3 Q4 & Q5

The reconstructed image is shown in Figure 13

Figure 13: Result of compressed image with only 6 components of the DCT transform

The image structure is preserved. However, there exists some notable blocking artifacts around edge details. The artifact is exacerbated by the fact that DCT blocks are independent of each other, as a result, the rendered block or abrupt and incoherent. In areas that were examined to be uniform, the thresholding had no effect – this is because of the aforementioned sufficient representation of the sub-image by low order frequencies. Features such as the hair, eyes, hat suffer from the most artifacts because they are areas that undergo larger changes in intensities. As a result, higher order frequencies are required to better represent that changes occurring in the sub-image. By masking high frequency terms to 0, the representation is lost, and edge detail is replaced by a low frequency smooth blob. This assimilates as individual blocky artificats in the final image.

4.4 Q6

In effect, what was achieved by the masking operation is a very aggressive quantization of the transformed data. This would be equivalent to setting the quality slider to the lowest possible. Nonetheless, the DCT works well at compressing images. Since sub-images tend to be uniform within a small 8x8 grid, low frequency components dominate. This allows the transform to neglect higher frequency terms during the quantization step. In other words, the transform is energy compact. Moreover, since the transform is represented by cosines, it is formed by real numbers; the resulting implementation is consequently easier and faster.

5 Quantization

Quantizations of increasing strength were performed over 4 iterations and are shown in progession in Figure 14.

Figure 14: Quantization scaled by 1, 3, 5, and 10

5.1 Q1

The quantization step simply divides and rounds each coefficient of the DCT by some strategic weighting that aims to truncate redundant component while preserving image integrity. By performing the division, the coefficient value becomes smaller, which in turn implies the frequency component has less contribution to the overall image. The goal is is trade off coefficients for quality - often be zeroing them. In other words, they must be divided by a large number.

Once said components are truncated, we lose representation of high frequency details. If the truncation is severe enough to truncate components we need to represent an image patch, we experience notable blocking artifacts – the sub-image is being approximated by lower order, smooth components.

5.2 Q2

With the quantization scaled by 3, the divisor grows larger. Some high frequency components with small coefficients are rounded to zero after the division – effectively nullifying their existence in the output image. When zooming into the image, we notice some blocking artifacts in the background and structural features as a result of the truncation.

5.3 Q3

The blocking artifact gets progressively worse as quantization is scaled. The quantization Z10 experiences the strongest artifact. This in turn detriments the PSNR and visual integrity of the image – the lossy compression is deviating pixel values from the true image. As the divisor gets larger, more harmonics are being filtered and rounded to 0. As a result, more structural representation is lost. The resulting image is represented by lower order harmonics that are perceived as more uniform, smooth, and blocky. The PSNR of each quantization scale is shown in Figure 15 and is observed to progressively decrease as quantization strength is increased.

5.4 Q4

Blocking artifacts are the most prevalent due to the loss of high frequency representation of structural details. As a secondary side effect of blocking, some ringing is observed around edges.

5.5 Q5

As formerly discussed in details, quantization is a trade-off between compression strength and quality. By increasing the quantization effect, higher order frequencies were truncated to zero, which nullified their existence in the final image. This greatly compresses the image as less components are required to reconstruct the image. However, beyond a certain

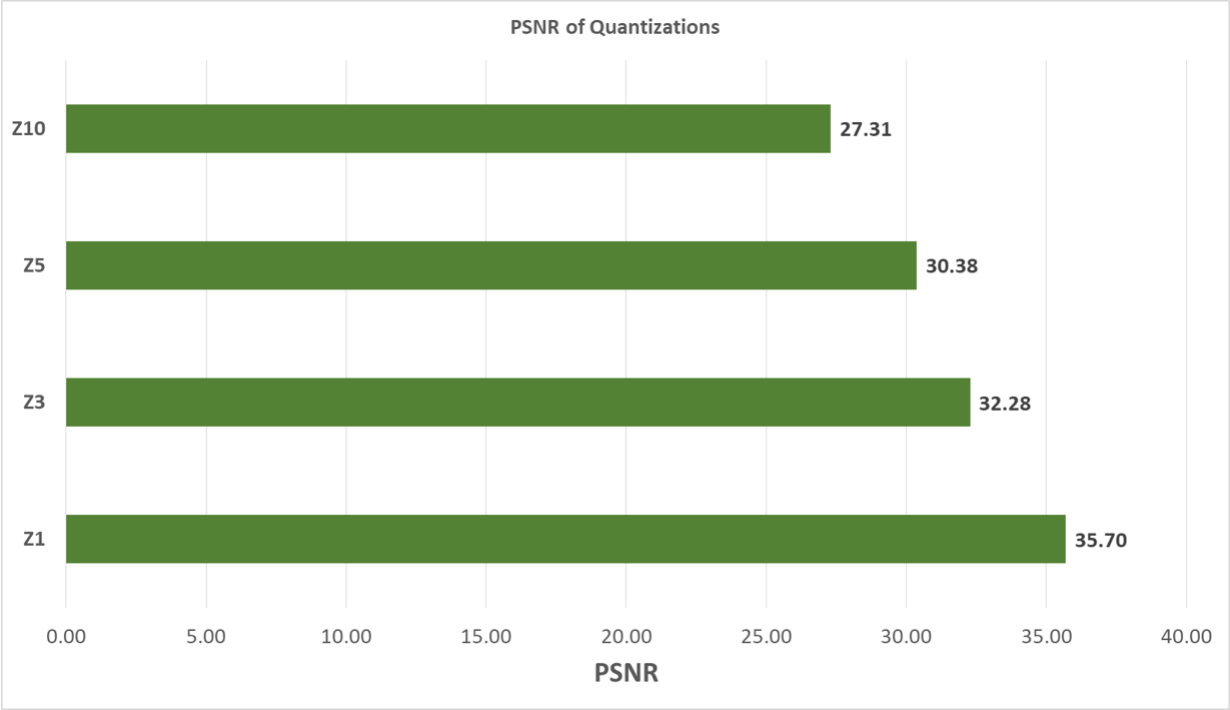

Figure 15: Quantization PSNR

threshold, we lose enough structural and edge details to the point where the human visual system becomes sensitive to the integrity loss. This is why quantization tables are specified by standards and vendors - they are strategically selected to truncate components that are less likely to exist in typical images. The JPEG standard's quantization table for instance is different than Photoshop's implementation of the quantization table.

6 Convolutional Neural Networks

6.1 Q1

MLP is more expensive in terms of parameters since it is fully connected - every perceptron is connected to another across layers, as a result, the number of parameters grow drastically. A CNN is more advantageous in image processing because it acts as a kernel that spans the image; as a result, it provides a spatial context that raw MLP inputs do not. The spatial context reduces redundancies in parameter count since they can be averaged or shared. This is possible because pixels in images are inherently correlated; hence, parameters can be related.

6.2 Q2

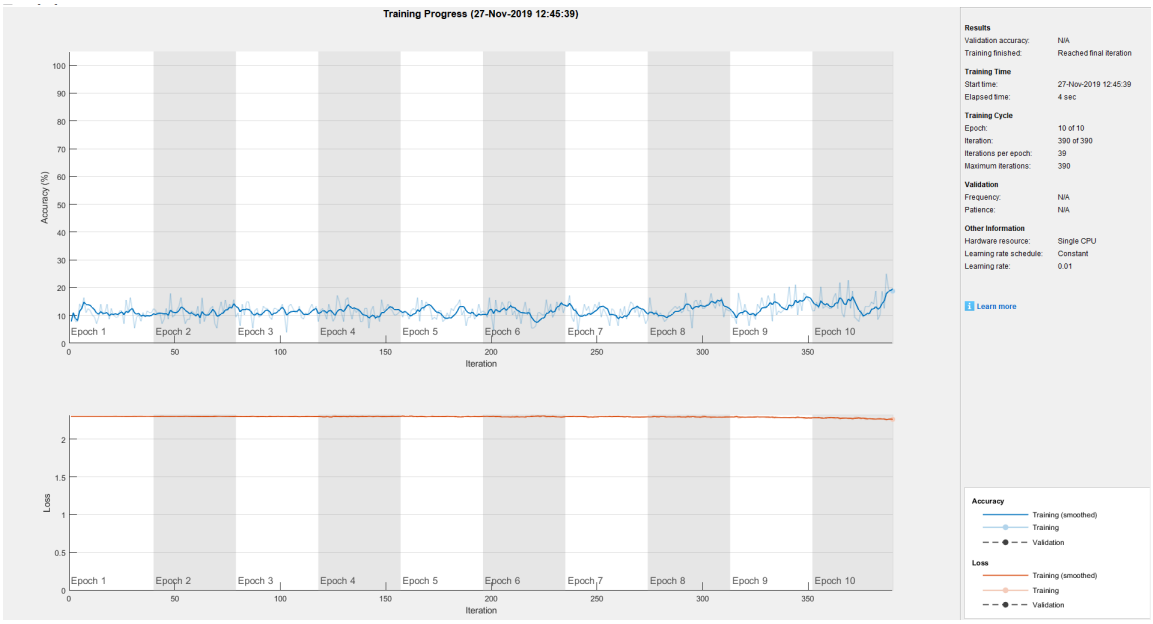

Figure 16: Training result of MLP. Epoch = 10, Rate = 0.01

The CNN approach takes an elapsed time of 17 seconds with a testing and training accuracy of 0.1135 and 0.1126 respectively. The MLP approach rapidly finishes in just 4 seconds with a testing and training accuracy of 0.2752 and 0.2772 respectively.

The MLP training is faster and converges to a more accurate value. After the 9th epoch, the accuracy of the model surges. Conversely, the CNN model converges to a lower accuracy with no abrupt changes after iterations.

It is possible that training an MLP model is faster because a larger number of parameters contains enough information to converge faster – even though CNNs are computationally more efficient.

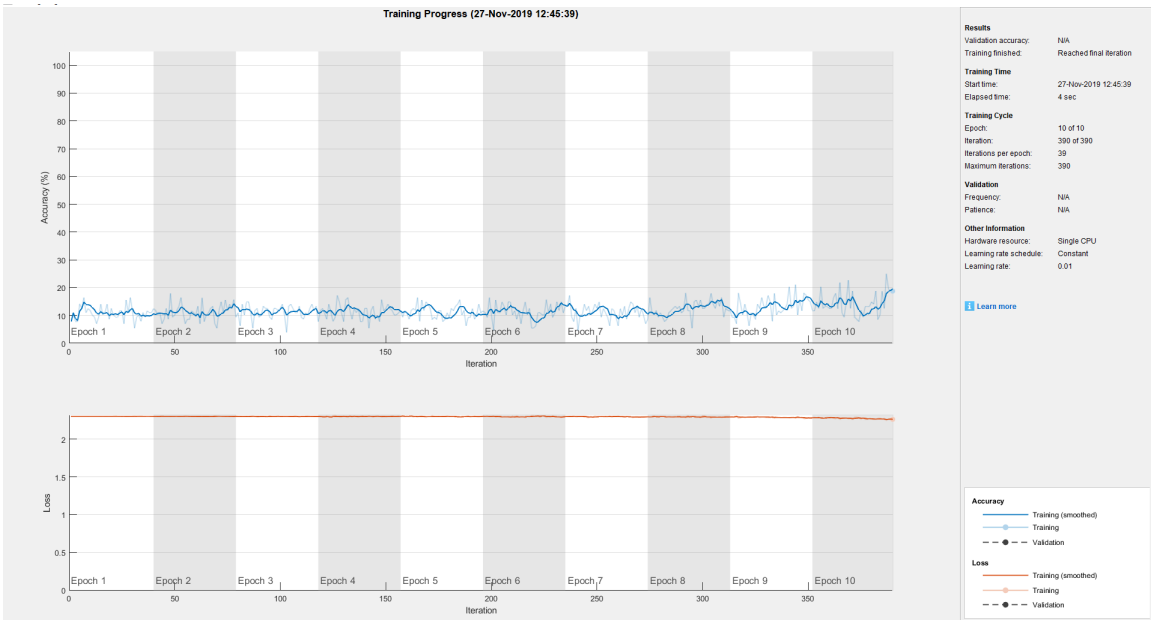

Figure 17: Training result of CNN. Epoch = 10, Rate = 0.01

6.3 Q3

In both models, the testing and training accuracy are similar. Testing accuracy is naturally lower since it examines the result on samples the models have not experienced or trained on previously.

6.4 Q4

By increasing the learning rate, the overall speed and accuracy of the training is unchanged in the CNN model.

On the other hand, the MLP model's speed remains the same, however the accuracy is drastically improved. Past the 2nd epoch, the accuracy exponentially grows in accuracy to reach as 0.9313 and 0.9992 for testing and training ac curacies respectively.

6.5 Q5

In the case of CNN, the accuracy begins surging past 55th epoch. The model immediately begins settling at a testing and training accuracy of 0.95 and 0.998 respectively. Since there is no discrepancies between training and testing results, there was no occurrence of over-fitting.

The MLP training finished in 37 seconds and appears to train significantly faster. It arrives at a convergent value past the 20th epoch. Eventually, a steady state value is reached, which yields a training accuracy of 1. However, the resulting testing accuracy falls back to 0.9301

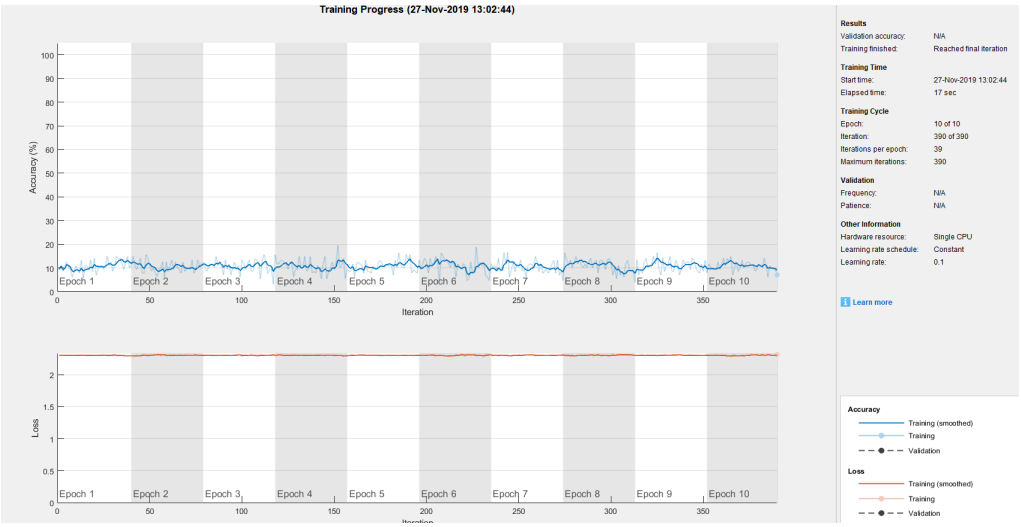

Figure 18: Training result of CNN. Epoch = 10, Rate = 0.1

- which is notable drop compared to the case of 10 epochs. With 10 epochs, the discrepancy was less than 0.7%, versus the 7% attained at 100 epochs. It is concluded that the model is susceptible to over-fitting.

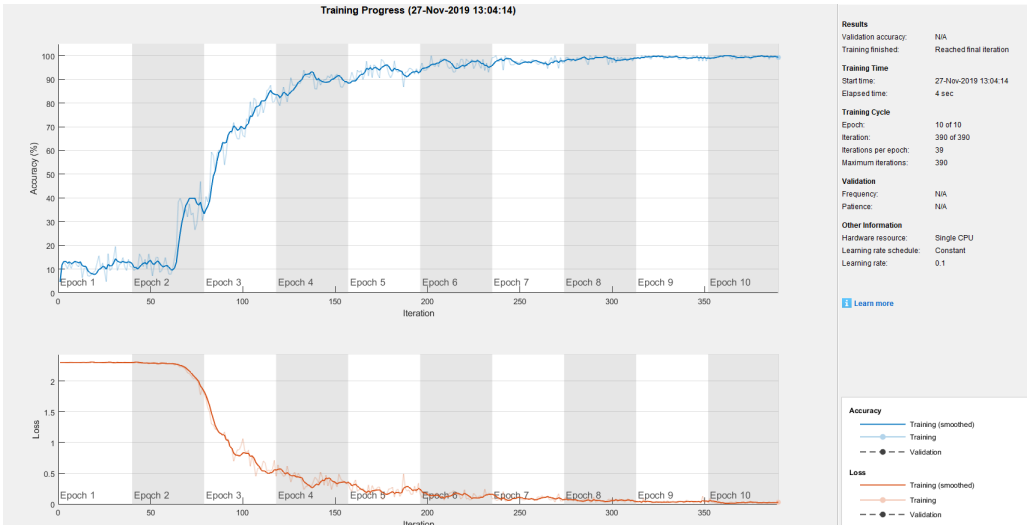

Figure 19: Training result of CNN. Epoch = 10, Rate = 0.1

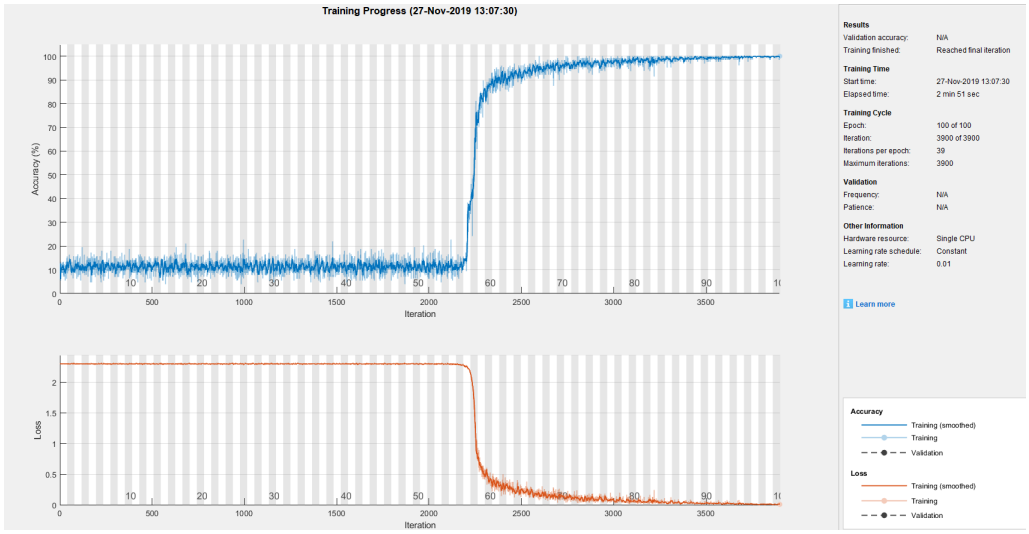

Figure 20: Training result of CNN. Epoch = 100, Rate = 0.01

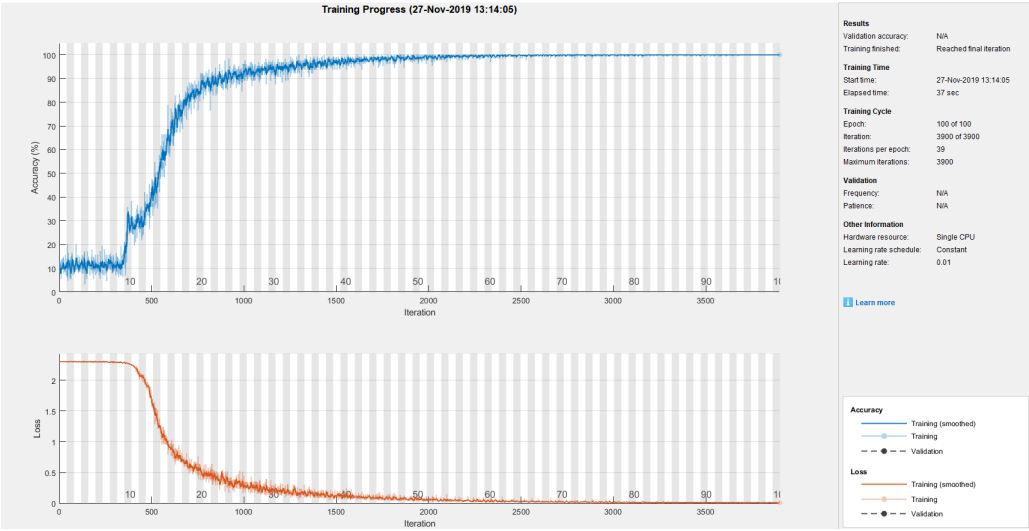

Figure 21: Training result of CNN. Epoch = 100, Rate = 0.01

7 Conclusion

The chroma sub-sampling process revealed the extent of irrelevance of color information to the human visual system. By initially reducing the redundancy found in RGB colors, the color information were further compressed by 75% with little to no impact on the integrity of the image. The same cannot be said when attempting to compress structural information such as the Y-channel of the YCbCr transformed image.

K-mean clustering is an iterative process that is capable of classifying images into arbitrarily sized clusters. This form of image segmentation is desirable due to its convergent solution in most scenarios. That is, the seed has little to no effect on the final result of the clustering. One exception that was noted is the possibility of uniformly color images; in that case, the choice of assigning classes can be arbitrary.

The DCT experiment revealed the relative importance of frequency components in the image. Out of 16 possible frequency composition of an 8x8 image patch, only the lower order harmonics are sufficient to represent the image with little loss in quality. A a transformation that removed all harmonics beyond the 6th yielded a sufficiently desirable image quality. The actual truncation of higher order components occur due to the division and rounding in the quantization step. By increasing the intensity of quantization, more notable lossy compression artifacts were identified – the compression is pushed to a limit where lower order components are not enough to represent features we care about.

The CNN vs MLP experiment highlighted the relevance of each model based on training time and accuracy. The MLP method was faster to train in all test cases. The over-parameterization inherent in the model can aid in producing a convergent solution faster than the CNN model. As a consequence of large parameter counts, the MLP method was susceptible to over-fitting over long training periods. The issue was not present in the CNN model due to its parameter sharing and spatial context.

8 Appendix

8.1 lab5.m

```
lena = imread('lena2.tiff');
peppers = imread('peppers.png');

%% 2. Chromatic Subsampling --done by Naji
peppers_ycbcr = rgb2ycbcr(peppers);

figure;
title('YCbCr');
imshow(peppers_ycbcr);

peppers_Y = peppers_ycbcr(:,:,1);
peppers_Cb = peppers_ycbcr(:,:,2);
peppers_Cr = peppers_ycbcr(:,:,3);

figure
    subplot(1,3,1)
        imshow(peppers_Y); % Y
        title('Y Channel')
    subplot(1,3,2)
        imshow(peppers_Cb); % Cb
        title('Cb Channel')
    subplot(1,3,3)
        imshow(peppers_Cr); % Cr
        title('Cr Channel')

% Downsize the chroma channels
%peppers_Y = imresize(peppers_Y, 0.5, 'bilinear');
ds_peppers_Cb = imresize(peppers_Cb, 0.5, 'bilinear');
ds_peppers_Cr = imresize(peppers_Cr, 0.5, 'bilinear');

% Upsize them again
ds_peppers_Cb = imresize(ds_peppers_Cb, 2, 'bilinear');
ds_peppers_Cr = imresize(ds_peppers_Cr, 2, 'bilinear');

figure
    subplot(1,2,1)
        imshow(cat(3, peppers_Y, ds_peppers_Cb, ds_peppers_Cr));
        title('Compressed CbCr Channels Result')
    subplot(1,2,2)
        imshow(cat(3, peppers_Y, peppers_Cb, peppers_Cr));
        title('Original Result')
```

```
% Downsize the Y channel
ds_peppers_Y = imresize(peppers_Y, 0.5, 'bilinear');
ds_peppers_Y = imresize(ds_peppers_Y, 2, 'bilinear');

figure
    subplot(1,2,1)
        imshow(cat(3, ds_peppers_Y, peppers_Cb, peppers_Cr));
        title('Compressed Y Channel Result')
    subplot(1,2,2)
        imshow(cat(3, peppers_Y, peppers_Cb, peppers_Cr));
        title('Original Result')

%% Section 3- K-MeansColour Segmentation-- Shilpan

C = makecform('srgb2lab');
converted_pepper = applycform(peppers,C);

ab = double(converted_pepper(:,:,2:3));
m = size(ab,1);
n = size(ab,2);
ab = reshape(ab,m*n,2);

%K = 2;
%row = [55 200];
%col = [155 400];
%mu = zeros(2,2);
 K = 4;
 row = [55 130 200 280];
 col = [155 110 400 470];
 mu = zeros(4,2);

% Convert (r,c) indexing to 1D linear indexing.
idx = sub2ind([size(converted_pepper,1) size(converted_pepper,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);
figure;
imshow(pixel_labels, []);
```

```
title('Image labeled by cluster index, K=4');
colormap('jet');

mask = repmat(pixel_labels, [1,1,3]);
clustered_peppers = peppers;

figure;
for k = 1:K
    temp = uint8(mask(:,:,1) == k);
    clustered_peppers(:,:,1) = peppers(:,:,1).*temp;
    clustered_peppers(:,:,2) = peppers(:,:,2).*temp;
    clustered_peppers(:,:,3) = peppers(:,:,3).*temp;
    subplot(2,2,k);
    imshow(clustered_peppers);
    title(sprintf('Segmented region %d',k));
end


%% 4. Image Transform (DCT)
f = rgb2gray(imread('lena2.tiff'));
f = double(f);
T = dctmtx(8);

imshow(T);

figure;
imshow(T);
title('8X8 DCT matrix');

% Apply transformation
F_trans = floor(blockproc(f-128, [8 8], @(x) T*x.data*T'));

figure
imshow(abs(F_trans), [])
title('DCT of entire image')

figure;
imshow(abs(F_trans(297:304, 81:88)), []);
title('8x8 F_trans at 297,81');

figure;
imshow(abs(F_trans(1:8, 1:8)), []);
title('8x8 F_trans at 1,1');

% Threshold
```

```
mask = [1 1 1 0 0 0 0 0;
1 1 0 0 0 0 0 0;
1 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0;
0 0 0 0 0 0 0 0];

F_thresh = blockproc(F_trans, [8 8], @(x) mask.*x.data);

f_thresh = floor(blockproc(F_thresh, [8 8], @(x) T'*x.data*T)) + 128;

figure;
imshow(f_thresh, []);
title('Final output after DCT transform');

lena_psnr = 10*log10(1/mean2((f-f_thresh).^2));

%% Section 5 - Quantization -- Shilpan
f = rgb2gray(imread('lena2.tiff'));
f = im2double(f);

Z = [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];

for i = [1,3,5,10]
    F_thresh = blockproc(F_trans, [8 8], @(x) round(x.data./(Z*i)));

    f_thresh = floor(blockproc(F_thresh, [8 8], @(x) T'*(x.data.*(Z*i))*T)) + 128;

    figure;
    imshow(f_thresh, []);
    title(sprintf('Quantized image with Z*%d', i));
    sprintf('PSNR of Z*%d: %d',i, psnr(f_thresh./255, f))
end
```