University of Waterloo

Faculty of Engineering

SYDE575 Lab 5: Image Compression and Segmentation

Prepared By
Sarthak Tamboli | 20665466
Sheen Thusoo | 20728766

Date: 7 December, 2021
Prepared for: Professor David A. Clausi

1. Introduction

The goal of this lab was to understand fundamental image compression concepts and techniques. This is important to learn as it helps in reducing storing and transmission bandwidth problems in digital graphic media. In section 2, we study the effects of chroma subsampling on image quality. In section 3, we studied colour segmentation; we explore *k*-means classification to segment various colours in an image using the RGB and *L*a*b** colour spaces. In section 4, we study the discrete cosine transform (DCT) which decomposes an image into a series of sinusoids with different frequencies and amplitudes. The characteristics of an image in the transform domain were studied. In section 5, we study the effects of different levels of quantization on image quality. Lastly, in section 6, we study multi-layer perceptrons (MLP) and convolution neural networks (CNN). We explore the usage of these networks to classify images of written digital from the MNIST dataset.

2. Chroma Sampling

In Section 2, the effects of chroma subsampling on image quality was studied. The peppers image was used and converted into the YCbCr colorspace. The original image is shown in Figure 1 and the images from each of the channels are shown in Figure 2. The code for this section is displayed in the Appendix in Figures 1.1.1 to 1.1.3.

Figure 1: Original Peppers Image

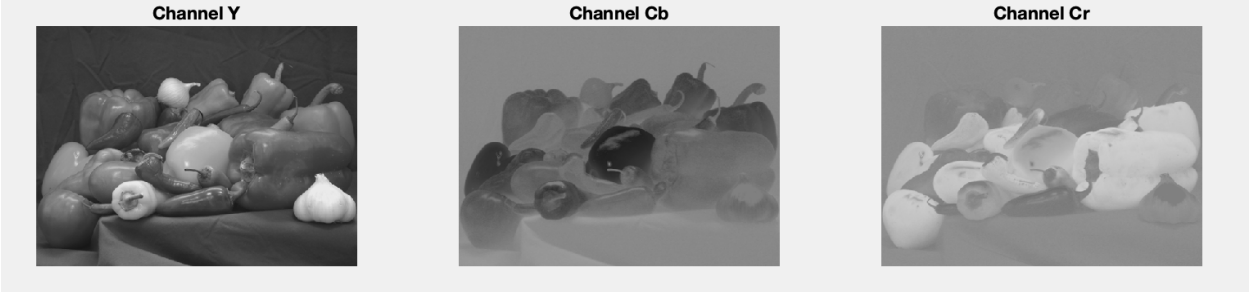

Figure 2: Plots of each image channel

1. From Figure 1, the Cb channel image seems to represent certain colours in the original image. It has darker pixels in the regions where the original image had yellow-orange pixels. This channel represents the blue difference or blue minus luma (B-Y). This means that pixels that are closer to a bluish colour in the original image are lighter intensities in the Cb channel image and colours that are more green/yellow get mapped to darker intensities. For example, the pepper in the middle is yellow (which is closer to red and green than blue) thus it shows darker intensities there. The Cr channel image looks brighter in regions where the original image was red (peppers). This is because the Cr channel represents the red difference or red minus luma (R-Y). This means that pixels that are closer to a red colour get mapped to lighter intensities on the Cr channel image. For example, the large red pepper on the right side of the image is a red colour and this gets mapped to lighter (whiter) intensities in the Cr channel.

2. Comparing the images in all three channels, the Y or luma channel contains more fine details. The Y channel image is a representation of the brightness of the original image; it represents the achromatic or grayscale image. This achromatic image contains more information about the fine details (edges) in the original image with more of the visual information retained. The Cb and Cr channels are chroma channels which contain information about the colours in the image. Since the human visual system is more sensitive to variations in brightness (luma) than colour (chroma), the Y channel is more important in retaining crucial information about the original image when reconstructing it.

Next, the resolution of the chrome channels were reduced by a factor of 2 in the horizontal and vertical directions. Then, these were upsampled back to the original resolution using bilinear interpolation. The color images were separated into three separate images so they can be operated on independently. The original Y channel image and the two upsampled Cb and Cr images were recombined to create a new colour image. The original and recombined images are shown in Figure 3.

Figure 3: Original Image and Chroma Subsampling Image

3. The resulting image from chroma subsampling and the original image, shown in Figure 3, look very similar to one another; there do not seem to be any large differences. In fact, there seems to be no visual differences that are noticeable to the human visual system.

4. Based on the resulting image, chroma subsampling seems not to have any noticeable effect on image quality. Thus, reducing the amount of chroma information stored compared to the amount of luma information has little impact on perceived image quality. This shows that chroma subsampling is an effective way to reduce image information while upholding image quality.

Next, the effects of luma subsampling were studied. The resolution of the luma (Y) channel was reduced by a factor of 2 in the horizontal and vertical directions. Then, it was upsampled back to the original resolution using bilinear interpolation. The upsampled Y channel image and original Cb and Cr images were then recombined to create a new colour image. The original and recombined images are shown in Figure 3.

Figure 4: Original Image and Luma Subsampled Image

5. Comparing the resulting image from luma subsampling with the original image, shown in Figure 4, the images seem to have a visual difference in image quality. The resulting image retains information about the colours and coarse details in the image; however, it looks much more blurry. The resulting image has blurry/smoothed fine details like edges. There seems to be a large visual difference in the images in terms of blurriness.

6. Based on the resulting image, luma subsampling reduces image quality. By reducing the resolution in this channel and then upsampling, we lose information about the fine details in the image. This results in a blurry image which has low image quality.

7. Comparing the resulting image from luma subsampling with the image produced using chroma subsampling, the latter method (chroma subsampling) performs better. The resulting

image produced from chroma subsampling had a significantly higher image quality than the luma subsampled image. The chroma subsampling method is better for reducing network bandwidth while preserving visual acuity. This is because it retained more of the fine details in the image like edges and looked very similar to the original image whereas the luma subsampled image looked very blurry and lost information about the fine details which are important to the human visual system.

3. Colour Segmentation

In Section 3, image segmentation was studied. Specifically, colour segmentation was explored using *k*-means classification to segment various colours in an image using the RGB and *L*a*b** colour space. This space consists of a luminosity dimension (*L**) and two colour dimensions called *a** and *b**. The *a** channel indicates colour which falls on the red-green axis and the *b** channel indicates colours which fall on the blue-yellow axis. The peppers image was loaded and converted to the *L*a*b** space. We then classified the colours in the *a** and *b** spaces using *k*-means clustering for $k = 2$ and $k = 4$. The code for this section is displayed in the Appendix in Figures A.2.1 to A.2.2.

Figure 5: Result of *k*-means clustering for $k = 2$

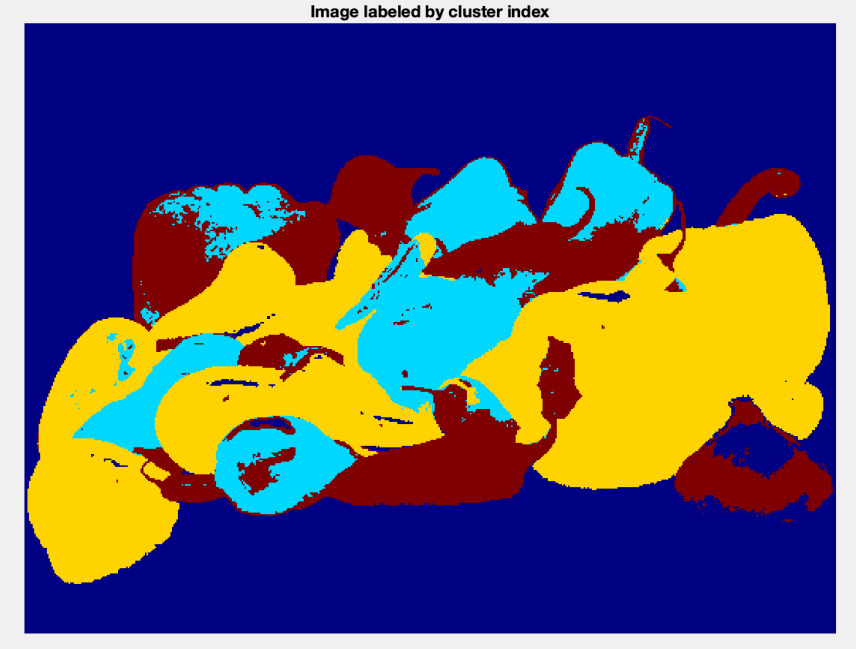

Figure 6: Result of *k*-means clustering for $k = 4$

8. Changing the value of *k* affected how the image was clustered. When *k* = 2, the image was clustered so that the background of the image was one cluster and another cluster was the peppers. In this case, the background pixels such as the purple fabric and some of the white pixels from the garlic were grouped into one cluster. The second cluster consisted of the red/yellow/orange peppers in the image. When we increased *k* to 4, more clusters were generated. Here, the different colours of pixels were grouped into different pixels. One cluster still consisted of the background (purple fabric) and some of the white pixels in the garlic. Another cluster grouped the orange and red peppers together. The yellow and light green regions of the peppers were grouped into one cluster and the last cluster consisted of the green peppers and some parts of the garlic.

9. In this case, the effect of the initial points on the final clusters was minimal. This is because the image was separated by intensities which are easier to distinguish. Thus, changing the initial points should result in similar output clusters. It does pose limitations on the speed of convergence. For instance, if the initial points are a large distance away from the optimal centroids, the algorithm will take more time to converge to these optimal clusters. This is due to the fact that this algorithm is iterative and works to compute new centroids until optimal points are found.

10.

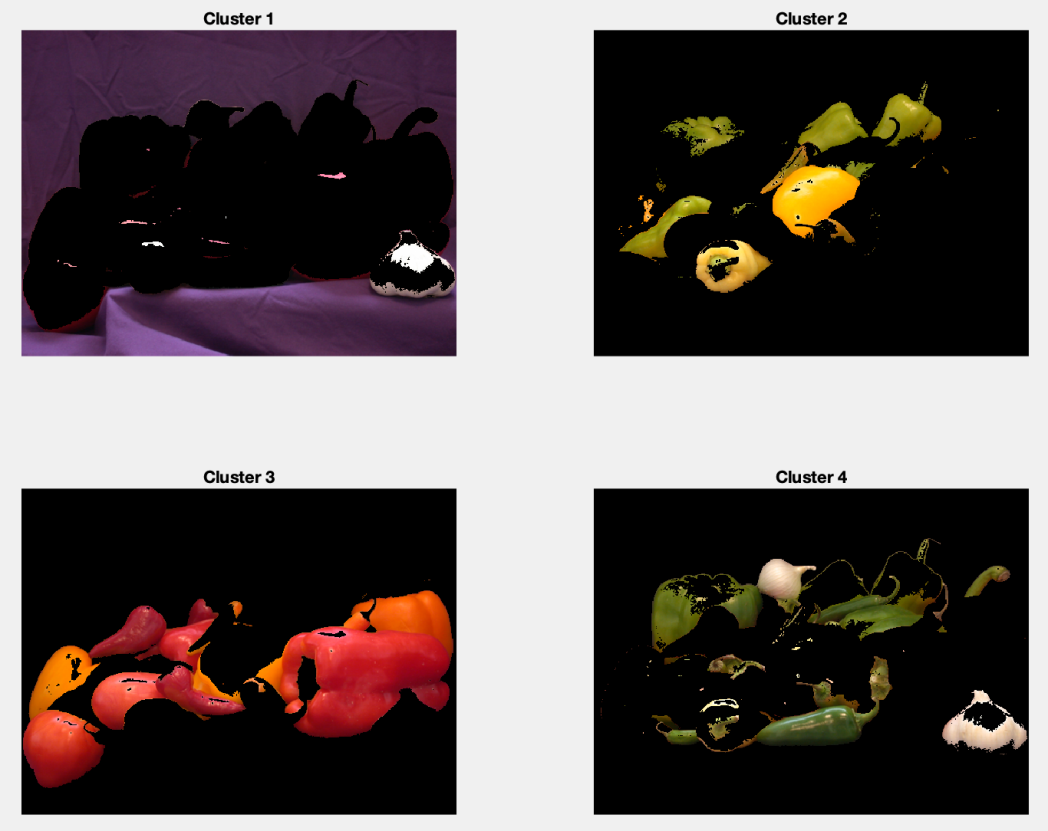

Figure 7: Image of each cluster for $k = 4$

The segmentation performed well for $k = 4$. The k-means algorithm was able to segment the image into different parts of the image. For example, one cluster grouped pixels belonging to the background (purple fabric) of the image and some parts of the garlic, one grouped pixels belonging to light green or yellow peppers, another grouped pixels belonging to red or orange pixels and the last grouped pixels belonging to the green peppers and some of the white garlic. From figure 7, cluster 1 and 4 were the ones where some pixels were grouped together even though they were not very similar. For example, cluster one includes pixels from the background and from some of the garlic and cluster four includes pixels from the green peppers and some parts of the garlic. This is probably due to the fact that only 4 clusters were generated; thus, the white garlic had to be grouped along with these even though it could be considered a separate part. If we increase k to 5 for example, we may see that the white garlic would be grouped into its own cluster.

4. Image Transform

In Section 4, the DCT and its characteristics were studied in the transform domain. A deeper dive was taken into DCT's and their pixel locations corresponding to energy distribution levels with respect to high and low frequencies. Their application in image compression was also explored. Finally, a reconstructed Lena image was generated with a PSNR value of 29.818079287248970. The relevant MATLAB code for this section can be found in Appendix A, Figure A.3.1, Figure A.3.2, and Figure A.6.1.

11.

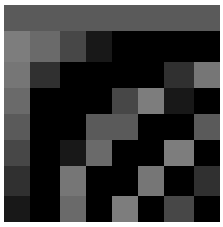

Figure 8: 8x8 DCT Transform Matrix

From Figure 8, it can be noted that each row constitutes cosines of different frequencies. From a top down view, we observe that the distribution is from low frequencies to high frequencies. The frequencies increase towards the lower rows. No frequencies belong to the first row, then a ramp and entire cycle are represented, in subsequent rows 2 and 3. A similar pattern follows till the bottom.

12.

Figure 9: Absolute Value of DCT Sub-Image w/ Top-Left at (81,297)

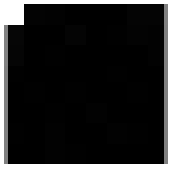

Figure 10: Absolute Value of DCT Sub-Image w/ Top-Left at (1,1)

By inspection of Figure 9 and Figure 10, an observation can be made that lower frequencies majorly comprise the energy, located near the top-left of the DCT sub-images. In these sub-image matrices the top-left pixel is representative of the smallest frequency magnitude for both vertical and horizontal directions. The magnitude of the highest frequency in both vertical and horizontal directions is stored on the bottom-right pixel. The top-right pixel determines the sensitivity to high frequencies for vertical edges. DCT is useful for image compression as most energy exists in lower frequencies, the higher frequencies can be omitted. Storing the low frequency information from the top-left pixel using encoding algorithms reduces the memory requirement of storing images. The Huffman encoding is one such algorithm that is currently employed.

13.

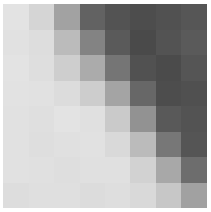

Figure 11: Original DCT Sub-Image w/ Top-Left at (81,297)

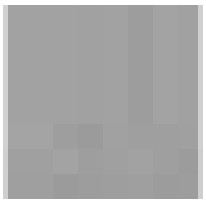

Figure 12: Original DCT Sub-Image w/ Top-Left at (1,1)

By inspection of Figure 9 and Figure 10 it can be said that the energy in Figure 9 is spread out across many pixels, in the top-left region, unlike Figure 10, where the majority if not all energy lies solely in the top-left pixel. To gain a deeper understanding the original DCT sub-images were plotted as showcased in Figure 11 and Figure 12. Figure 11, the DCT sub-image with top-left at (81,297) has visible intensity changes at a diagonal. The change in intensity can be correlated to an edge that occurs in the image, which is more vertical than horizontal. As the change is considerable, the impact will be more prevalent in the low frequencies. Furthermore the vertical components will be affected more than the horizontal due to the orientation of the diagonal intensities. From Figure 12, there are no considerable intensity changes observed; energy lies in low frequencies and this is evident in the top-left pixel from Figure 10.

14.

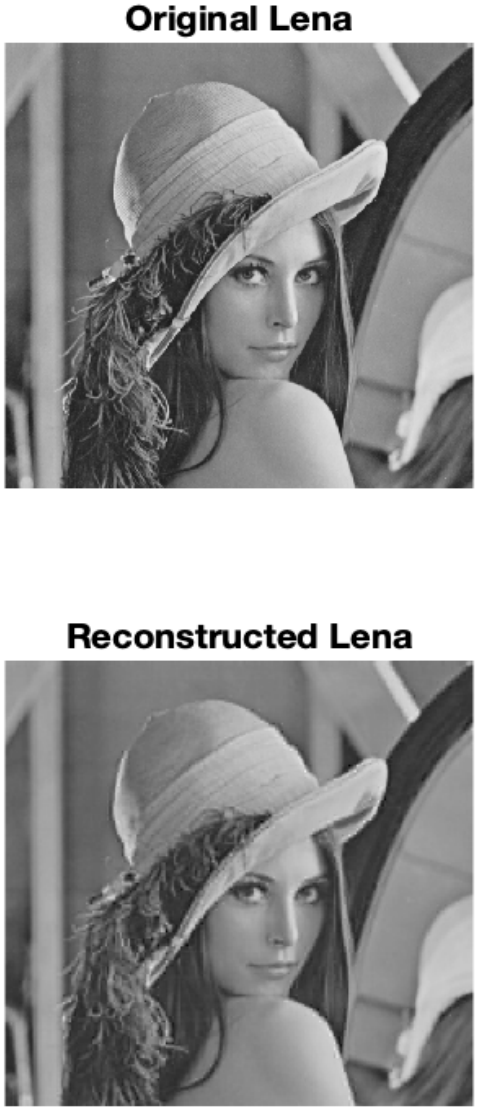

Figure 13: Original Lena (top) versus Reconstructed Lena (bottom)

Figure 13 showcases the original (top) image and the reconstructed (bottom) image. The reconstructed image has a PSNR value of 29.818079287248970. The reconstructed image looks visually very similar to the original image however, with less definition and sharpness and more overall smoothness. Specific locations of the image where this can be observed is Lena's hat, hair and her shoulder/arm. The aforementioned locations experience a degradation in sharp edges and definition as it appears smoothened. This effect is observed as the high frequencies from the original image have been removed. During the reconstruction, only 6 DCT coefficients were retained, which all happened to be related to low frequencies as they were located in the top-left region of the matrix. Recall, high frequencies contain detailing information such as edges and noise information; as this is removed edges are now smoothed and subsequent effects are observed.

15. The most prominent artifact in the image is a blocking artifact. The DCT Transform Matrix was of size 8x8 and the same pattern appears in the reconstructed image. Blocks of size 8x8 can be observed in the bottom image of Figure 13, that are different from their neighbouring 8x8 blocks. As each 8x8 block from the original image was constructed independently by the DCT Transform Matrix, a blocking artifact appears as a product of this process.

16. The DCT approach for image compression works extremely well. An 8x8 DCT Transform Matrix was constructed for image reconstruction. In total there were 64 DCT coefficients spanning all frequencies, high and low. From the 64, only 6 coefficients were used to reconstruct the image as shown in Figure 13 that produced a considerably well PSNR value of 29.818079. Only 9.375% of the original image data was required to be stored for a reconstruction that is adequate for the HVS. The removal of the majority of image data is only acceptable as most information removed is high frequency detailing that isn't easily perceivable by the HVS.
Thus, despite the omission of 58 DCT coefficients and high image compression the reconstructed image is highly similar.

5. Quantization

Quantization methods in image compression were explored in Section 5, to mitigate unwanted artifacts from appearing. The effects of different quantization levels with their related PSNR values are listed below in Table 1. The grayscale version of the Lena image was used as the reference image. The relevant MATLAB code can be found in Appendix A, Figure A.4.1 and Figure A.6.1.

Table 1: PSNR Values for Section 5

Quantized Image with Z Factor of	**PSNR Value**
1Z	35.731416966395614
3Z	32.314432656032245
5Z	30.413220298973584
10Z	27.344273308872506

Figure 14: Reconstructed, quantized, Lena image w/ 1Z Quantization Matrix

Figure 15: Reconstructed, quantized, Lena image w/ 3Z Quantization Matrix

Figure 16: Reconstructed, quantized, Lena image w/ 5Z Quantization Matrix

Figure 17: Reconstructed, quantized, Lena image w/ 10Z Quantization Matrix

17. The original quantization matrix has values increasing on a cascading diagonal from top-left to bottom-right. Quantization divides the sub-image by the matrix itself both rounded and element-wise. As the high frequency values are larger than the low frequency values after quantization they usually tend towards 0, after division in particular. This pattern results in the high frequency detailing including edges and noise to be removed from the reconstructed image. This removal leads to an overall smoothing effect visible throughout the image. In terms of the PSNR values and performance for the HVS, it still performs exceedingly well. The HVS is not receptive to high frequency detailing as compared to low frequency details. As the quantization levels are increased, the number of low frequencies removed also increases. The removal of low frequencies produces the blocking artifact in smoothened locations, where intensity values change slowly.

18. In Figure 15, the 3 level quantization reconstructed image is shown. Compared with Figure 14, a single quantized level reconstruction and Figure 13 (top), original Lena image, it can be said that the 3Z image is blurry with prominent blocking artifacts. As the quantization levels are increasing, a larger number of low frequencies are being removed during the compression process and that leads to blocking occurring for each sub-image. With respect to the original image, since the original image has all high and low frequency information, the sharp edges and noise are evident and exist so the 3Z image is definitely more pixelated in comparison. In regards to the 1Z image from Figure 14, the 3Z image experiences three times the quantization, the smoothening and blocking.

19. The reconstructed images are showcased in Figure 14, Figure 15, Figure 16, and Figure 17, starting from 1Z quantization to 10Z quantization. The related PSNR values are showcased in a tabular format in Table 1, at the beginning of this section. It can be observed that as the increasing the quantization levels leads to worse image quality both visually and according to the PSNR value. The maximum PSNR value is 35.731416966395614, for the 1Z level and the minimum is 27.344273308872506, for 10Z. The 3Z PSNR value is approximately 32.314432656032245 and the 5Z PSNR value is 30.413220298973584. The blocking artifacts become increasingly prominent as the quantization levels increase. This is due to the increased number of low frequencies being removed and rounded to 0 as they are located near the top-left region of the DCT. The result is a reconstruction with minimal low frequencies.

20. The blocking artifact becomes more prominent as the level of quantization increases. This is observed both quantitatively and qualitatively. Figure 14 to Figure 17, showcases the image degradation as quantization increases due to the increased prominence of the blocking artifacts throughout the image. The artifacts occur the most during the smoother edges and regions where intensities change gradually. Furthermore, a larger quantization level removes a larger number of DCT coefficients which results in a smaller amount of low frequency information to reconstruct the image. In theory, if the image was reconstructed with only the top-left pixel frequency information then it would resemble a flat image. This tendency can be observed in Figure 16 and Figure 17, as the higher quantization levels increase the pixelation, the blocking artifacts and overall smoothness of the image with greater loss of low frequency information.

21. Overall, it can be said that there is a trade-off between the quantization levels for reconstructing a compressed image. With lower quantization levels such as 1 and 3, it can be observed from Figure 14 and Figure 15 that the resulting images are acceptable with high performing PSNR values. The effects of the blocking artifacts are not as pronounced as the higher quantization levels however, still exist in comparison to the original image. The trade-off here is that the reconstruction is done with the high frequency image information as most of the DCT coefficients are omitted, except for the top-left regions information. Figure 14, is very similar to the original image and the loss of the high frequency information is not easily evident by the HVS. The quantization process is designed to divide low frequency information by small values to preserve the information and divide high frequency information to later round it off to 0, thereby eliminating the information. Since the HVS's sensitivity to high frequency information is low, the difference can be deemed negligible for low quantization levels. Considering high quantization levels, a larger number of DCT coefficients are removed, which inevitably starts removing low frequency information. Recall low-frequency information is the coarse image data that exists. The removal of the perceived, low, frequencies, in regards to the HVS, produces a worse image which is also indicative in the PSNR values. Furthermore, the compression performance does increase as less frequency information is stored and required, with the trade-off of poor image quality. Lastly, the degradation of image quality also produces a blocking

artifact that exists more prominently across the image as the quantization level is increased. Therefore, it can be said that the quantization process is effective if the quantization level is low and does not remove enough low frequency information to cause blocking artifacts, pixelation or blurriness. In terms of the performance even with a low quantization level, most of the DCT coefficients are not stored as most of the energy lies in the top-left low frequency region of the DCT image, so the process is memory efficient. The PSNR values for low quantization levels are also superior to that of high quantization levels.

6. Convolutional Neural Networks

In section 6, two types of neural network model were explored: multi-layer perceptron (MLP) and convolutional neural network (CNN). Here, the number of epochs and learning rate were tuned.

22. In terms of number of parameters, a fully connected layer is more expensive than a convolutional layer. In a fully connected layer, each node in the current layer uses a linear combination of each element in the previous layer. The number of total parameters can grow to be very high since each node in each layer is connected. For convolutional layers, the weights are shared and the layers are partially connected; every node does not connect to every other node. In terms of runtime, a convolutional layer is more expensive. Convolutional layers are beneficial for images because they represent an image array with dimensions of width, height, and channel; thus, they are able to retain spatial information. Fully connected layers are not able to retain spatial information.

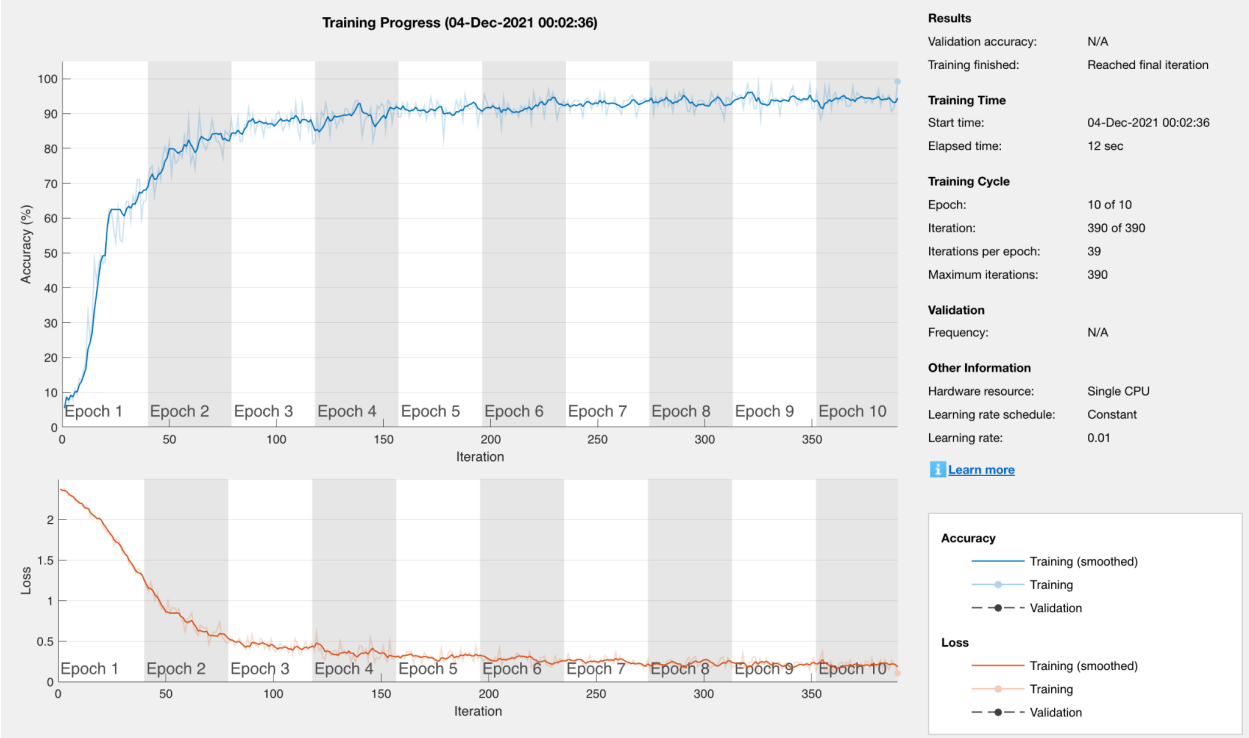

Figure 18: MLP Network with learning rate of 0.01 for 10 epochs

Test Accuracy: 0.9050
Training Accuracy: 0.9464

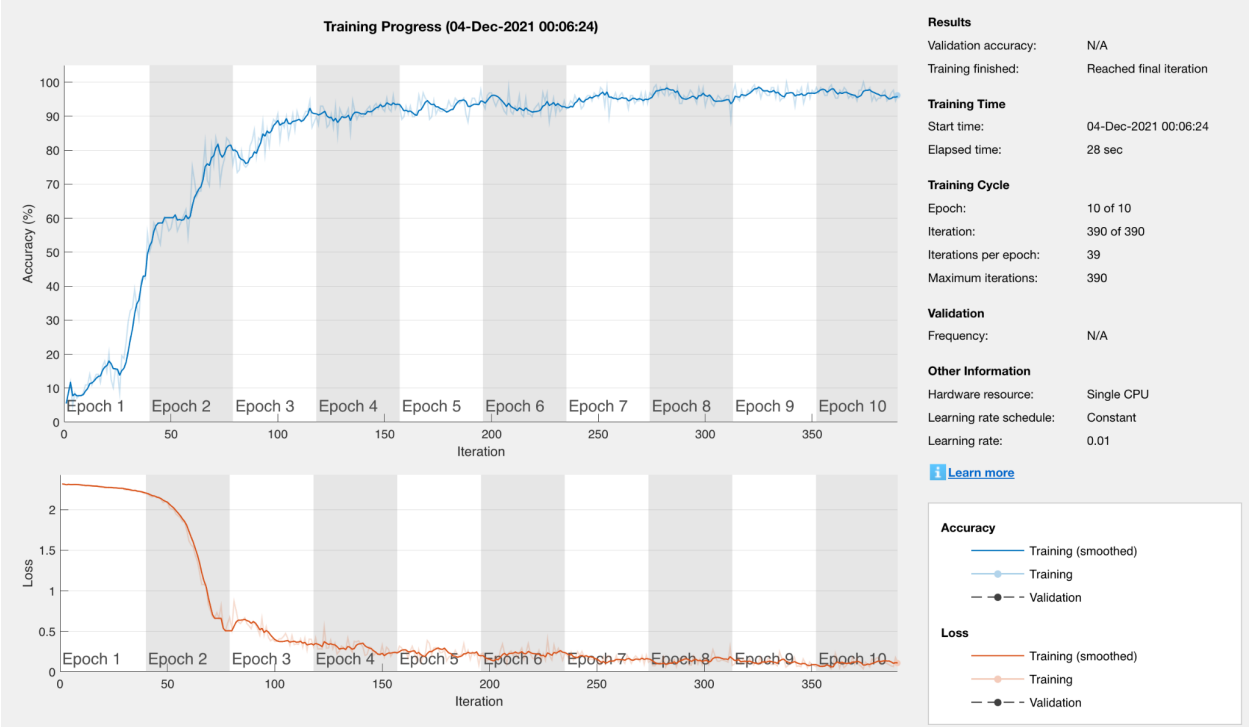

Figure 19: CNN with learning rate of 0.01 for 10 epochs

Test Accuracy: 0.9503
Training Accuracy: 0.9720

23. From figures 18 and 19, it is seen that initially, the accuracy of MLP shoots up to above 85% after epoch 2 versus CNN whose accuracy is around 80% after epoch 2. However, after these initial epochs, the accuracy of the CNN graph (around 95% after epoch 5) surpasses that of the MLP graph (around 90% after epoch 5). Thus, it seems that the CNN trains quicker than the MLP model. After epoch 10, both models seem to be at a similar error rate around 0.25; however, neither of them converge. The CNN model had higher training and test accuracies of 97.2% and 95.03%, respectively. Comparatively, the MLP model had lower training and test accuracies of 94.64% and 90.5% respectively. From the graphs, neither of the models converged.

24. The test accuracy is similar to the training accuracy, but it is always lower. This is because the model is trained on the training set, so it is able to classify the data better (it has seen it before). It is less accurate on the test set as it has never seen this data before.

25.

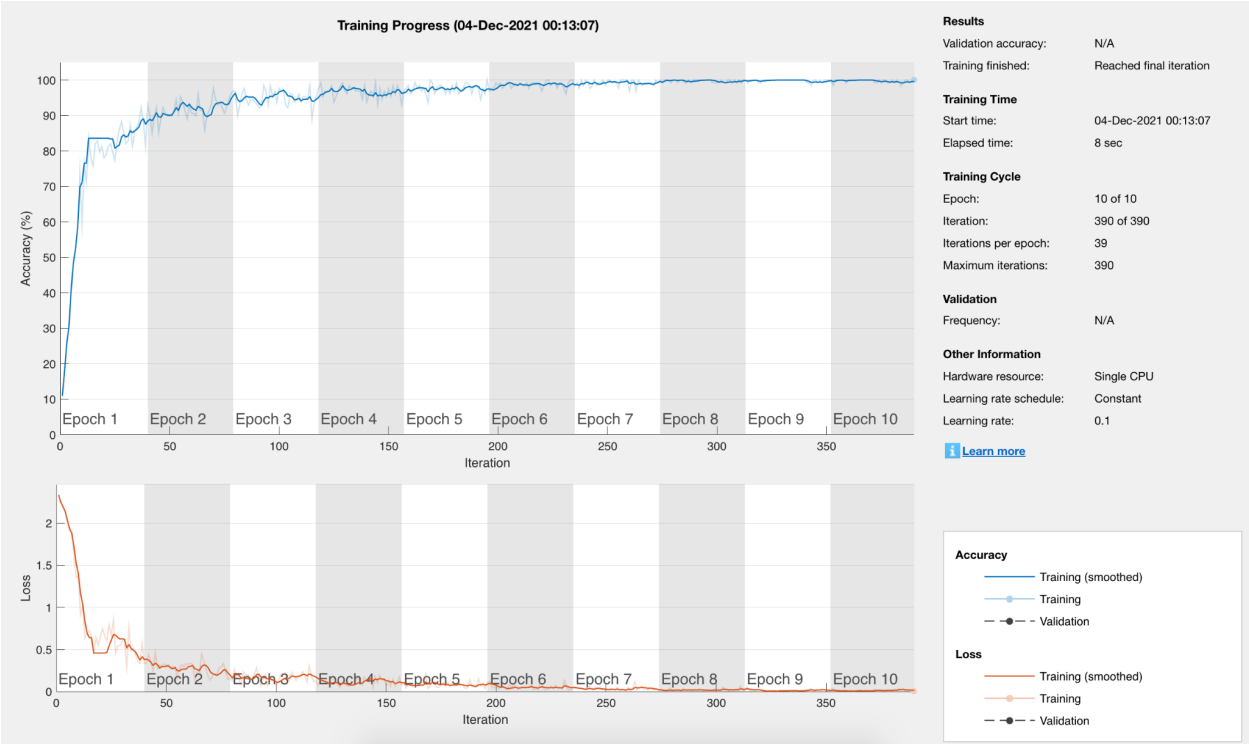

Figure 20: MLP Network with learning rate of 0.1 for 10 epochs

Test Accuracy: 0.9368
Training Accuracy: 0.9994

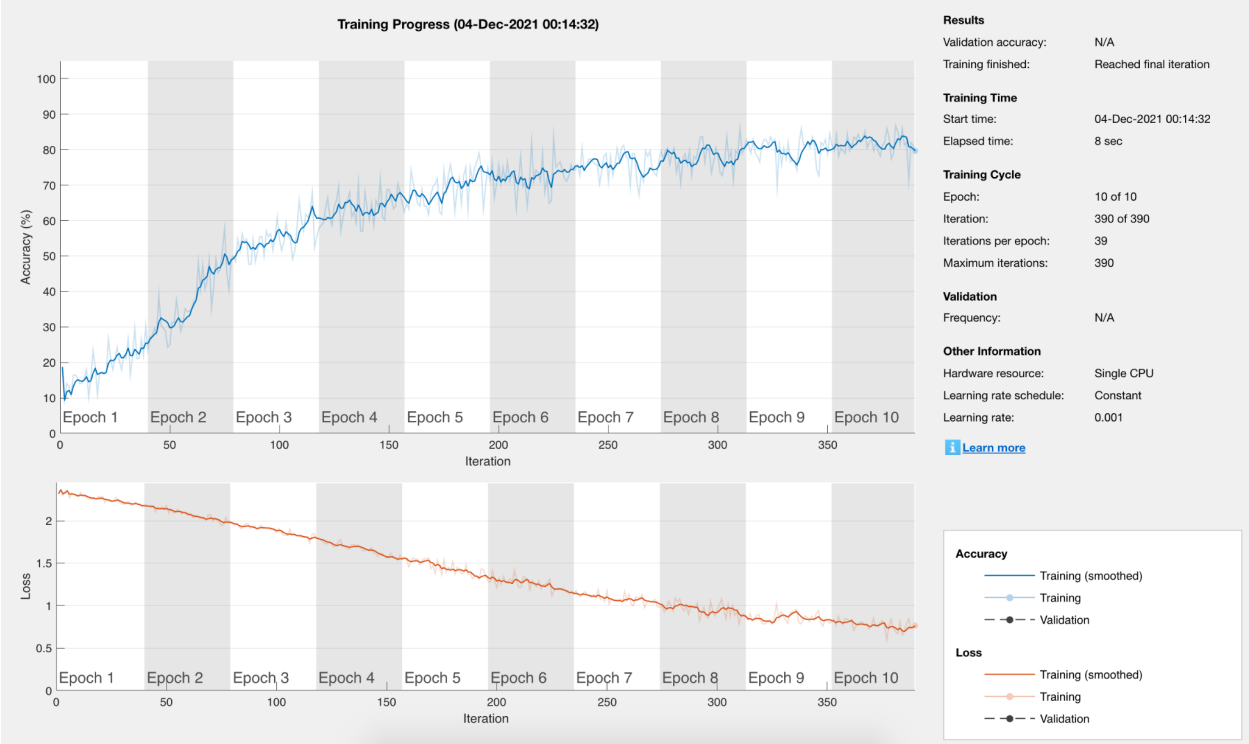

Figure 21: MLP Network with learning rate of 0.001 for 10 epochs

Test Accuracy: 0.8164
Training Accuracy: 0.8218

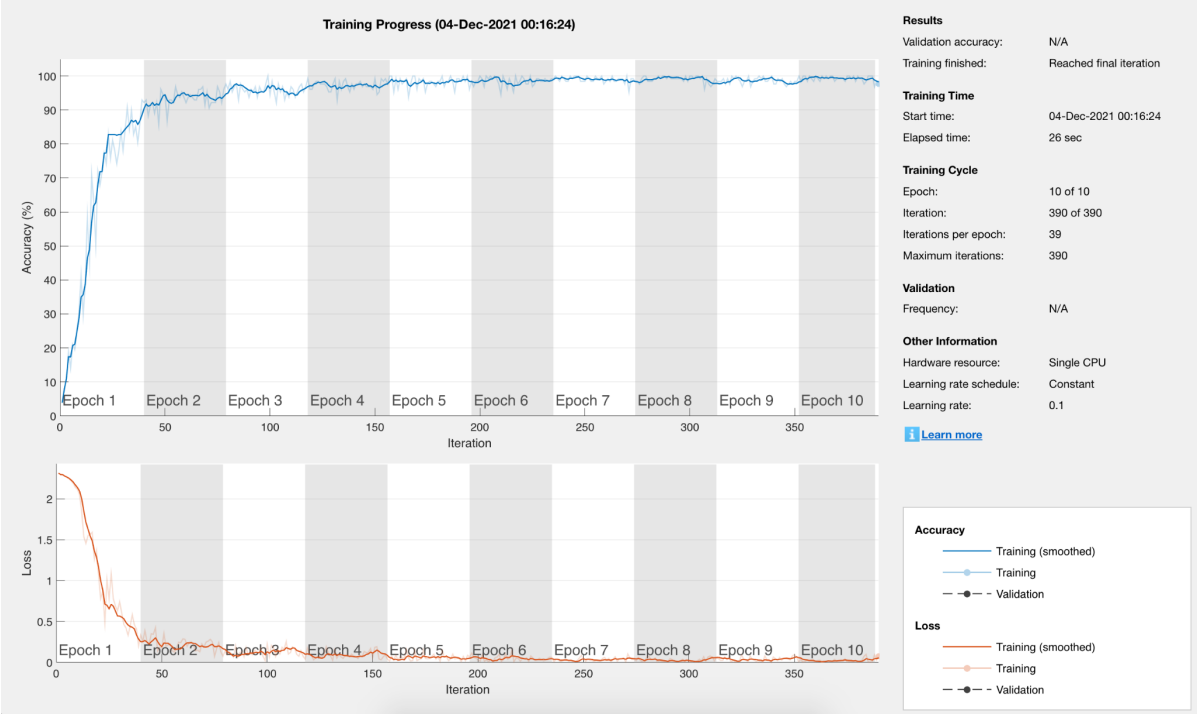

Figure 22: CNN with learning rate of 0.1 for 10 epochs

Test Accuracy: 0.9672
Training Accuracy: 0.9948

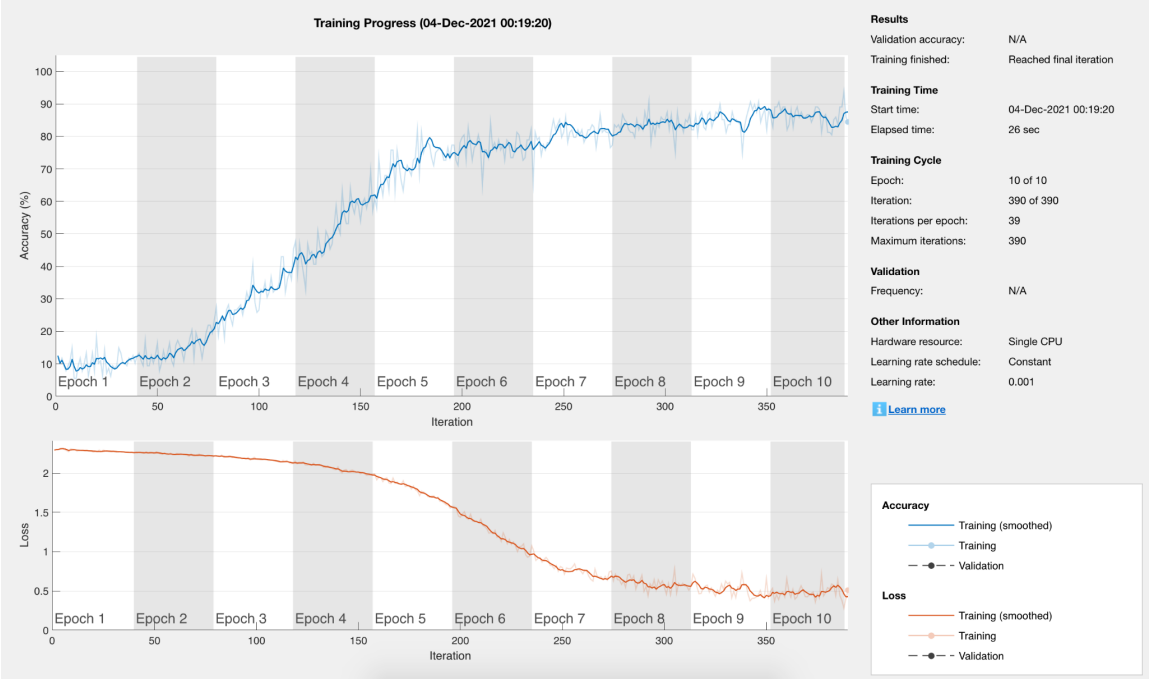

Figure 23: CNN with learning rate of 0.001 for 10 epochs

Test Accuracy: 0.8675
Training Accuracy: 0.8738

When both models are set to a learning rate of 0.1, the speed of convergence is faster than that with a learning rate of 0.001. When the learning rate is 0.1, the models converge by around epoch 3. However, when the learning rate is set to 0.001, neither of the models converge within 10 epochs.

When the learning rate is 0.1, the training graphs shoot up very quickly after epoch 1, and increase a bit before levelling off for the rest of the epochs. The graphs seem a bit more stable as there are not many jumps. When the learning rate is 0.001, the training graphs gradually increase taking 5 epochs to reach 80% accuracy. The graphs seem more unstable as the values jump up and down more.

26.

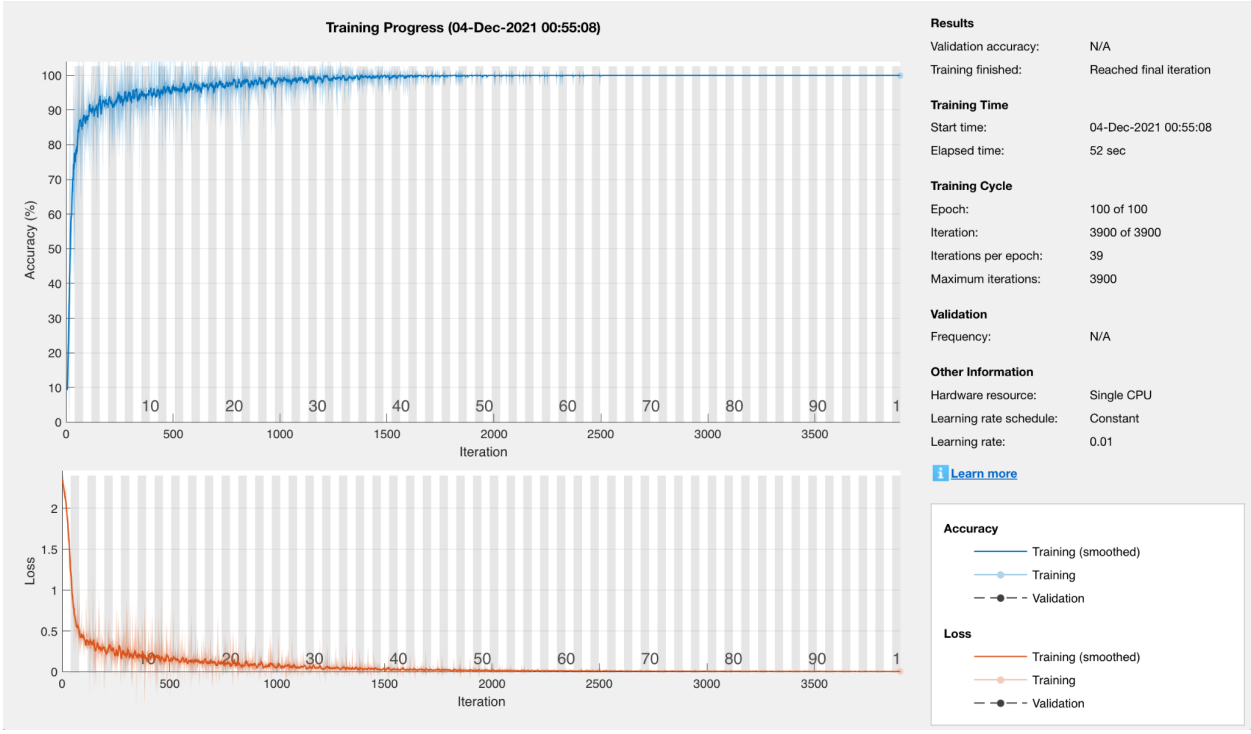

Figure 24: MLP Network with learning rate of 0.01 for 100 epochs

test_accuracy =0.9341
train_accuracy =1

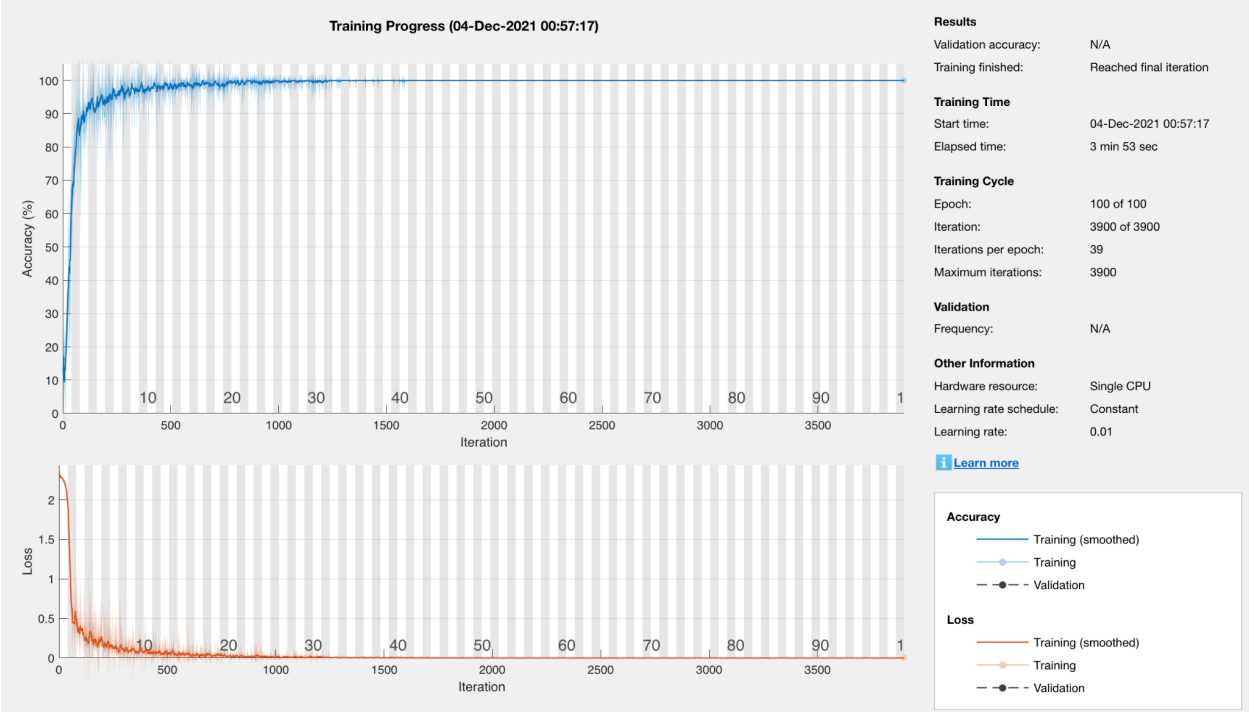

Figure 25: CNN with learning rate of 0.01 for 100 epochs

test_accuracy = 0.9716
train_accuracy = 1

For both networks, when trained for 100 epochs, overfitting occurs. This is clearly seen as the train accuracy for both is at a full 100%. The test accuracies for these models are also high at values at 93% and 97% accuracy; however, this could be due to similarities in data between the training and test sets. Since the models have trained to be 100% accurate on the training set, it is likely that when the model sees new data, it will not be able to accurately classify it. The reason overfitting occurred was due to the large amount of epochs it was trained for. When only trained for 10 epochs, the models do not overfit the data (the training accuracies are lower than 100%).

7. Conclusion

In section 2, the effects of chroma and luma subsampling on image quality was studied on the peppers image. It was found that the luma or Y channel contained more information on fine details and image structure. Therefore, the luma channel is important in reconstructing the image. Reducing the resolution of the chroma channels did not seem to affect the image quality; however, reducing the resolution of the luma channel greatly reduced the image quality by making the image more blurry. Thus, reducing the amount of chroma information stored compared to the amount of luma information has little impact on perceived image quality.

In section 3, colour segmentation was explored using k-means classification to segment various colours in an image using the RGB and L*a*b* colour space. Changing the value of k from k=2 to k=4 changed the way the image was clustered. When k=2, the image was segmented into two regions, the background of the image and the objects in the image (the peppers). When k=4, the image was segmented according to the different colours of objects in the image. For example, the green peppers were one cluster and the orange/red peppers were grouped into another cluster. The algorithm may have been better able to segment the garlic from the image if another cluster was added (k=5).

In section 4, an 8x8 DCT Transform Matrix was used to compress the Lena image. It was observed that most of the energy was located in the top-left pixel, or low frequency region, of the DCT. Detailing and definitions including edges and noise exist in the high frequency regions. Removing the majority of high-frequency, more specifically 58 DCT coefficients out of 64, retaining only 6 DCT coefficients located in the top-left region performed well in the reconstruction of the image. The resulting PSNR value was 29.81. Blocking artifacts did become prominent in sizes of 8x8 blocks due to the 8x8 DCT Transform Matrix however, the effect of the edge smoothness and loss of high frequency detailing is acceptable in accordance with the HVS as it is not very susceptible to high frequency information.

In section 5, varying quantization levels were applied to Lena. Their effects on the reconstructed image, compression performance, PSNR values and HVS interpretation were analyzed. It was deemed that it's effective and the nature of quantization matrices is to retain low frequencies and round off high frequency information to 0, producing a clear trade-off. Increasing the quantization level reduces the low frequency information stored which increases the prominence of blocking artifacts and degrades the image quality. The best performing quantization was 1Z with a PSNR value of 35.73 and the worst was 10Z with a value of 27.34.

In section 6, two types of neural network model were explored: multi-layer perceptron (MLP) and convolutional neural network (CNN). It was found that CNNs are better for an image classification task as they take into consideration spatial information. The learning rates and number of epochs for both of these models was tuned. When the learning rate was increased by a factor of 10 to 0.1, the models converged more quickly. Furthermore, when the number of epochs was increased to 100, both of the models experienced overfitting where they had training accuracies of 100%. Thus, it is important to tune the learning rate and number of epochs appropriately so that the model performs successfully.

Appendix A

```
% convert the image from the RGB colorspace into the YCbCr colorspace
% using the rgb2ycbcr function
peppers_rgb = imread('peppers.png');
peppers_ycbcr = rgb2ycbcr(peppers_rgb);

figure;
imshow(peppers_rgb);

figure;
subplot(1,3,1), imshow(peppers_ycbcr(:,:,1));
title('Channel Y');
subplot(1,3,2), imshow(peppers_ycbcr(:,:,2));
title('Channel Cb');
subplot(1,3,3), imshow(peppers_ycbcr(:,:,3));
title('Channel Cr');

% Now, reduce the resolution of the chroma channels by a factor of 2 in both
% the horizontal and vertical directions and then upsample them back to the
% original resolution using bilinear interpolation

cb_upsampled = imresize( imresize(peppers_ycbcr(:,:,2), 0.5), 2, 'bilinear');
cr_upsampled = imresize( imresize(peppers_ycbcr(:,:,3), 0.5), 2, 'bilinear');

```

Figure A.1.1: Code for Section 2

```
% Recombine the original Y channel image and the two upsampledCb and Cr
% images to create a new color image
ycbcr_upsampled = peppers_ycbcr;
ycbcr_upsampled(:,:,2) = cb_upsampled;
ycbcr_upsampled(:,:,3) = cr_upsampled;

recombined = ycbcr2rgb(ycbcr_upsampled);

figure;
subplot(1,2,1), imshow(peppers_rgb);
title('Original Image');
subplot(1,2,2), imshow(recombined);
title('Chroma Channel Resized Image');

% Reduce the resolution of the luma (Y) channel by a factorof 2 in both the
% horizontal and vertical directions and then upsample it back to the origina
upsampled_y = imresize( imresize(peppers_ycbcr(:,:,1), 0.5), 2, 'bilinear');

% Recombine the upsampled Y channel image and the original Cb and Cr images
% to create a new color image
y_resized = peppers_ycbcr;
y_resized(:,:,1) = upsampled_y;
```

Figure A.1.2: Code for Section 2 (continued)

```
42      % Recombine the upsampled Y channel image and the original Cb and Cr images
43      % to create a new color image
44 -    y_resized = peppers_ycbcr;
45 -    y_resized(:,:,1) = upsampled_y;
46
47 -    recombined_y = ycbcr2rgb(y_resized);
48
49 -    figure;
50 -    subplot(1,2,1), imshow(peppers_rgb);
51 -    title('Original Image');
52 -    subplot(1,2,2), imshow(recombined_y);
53 -    title('Y Resized Image');
```

Figure A.1.3: Code for Section 2 (continued)

```
 1 -    peppers = imread('peppers.png');
 2 -    peppers_lab = applycform(peppers, makecform('srgb2lab'));
 3
 4 -    ab = double(peppers_lab(:,:,2:3)); % NOT im2double
 5 -    m = size(ab,1);
 6 -    n = size(ab,2);
 7 -    ab = reshape(ab,m*n,2);
 8
 9      % K = 2;
10      % row = [55 200];
11      % col = [155 400];
12 -    K = 4;
13 -    row = [55 130 200 280];
14 -    col = [155 110 400 470];
15
16      % Convert (r,c) indexing to 1D linear indexing.
17 -    idx = sub2ind([size(peppers,1) size(peppers,2)], row, col);
18      % Vectorize starting coordinates
19 -    for k = 1:K
20 -        mu(k,:) = ab(idx(k),:);
21 -    end
22
23 -    cluster idx = kmeans(ab, K, 'start', mu);
```

Figure A.2.1: Code for Section 3

```
24
25      % Label each pixel according to k-means
26 -    pixel_labels = reshape(cluster_idx, m, n);
27 -    h = figure, imshow(pixel_labels, []);
28 -    title('Image labeled by cluster index');
29 -    colormap('jet')
30
31      % Output each cluster/segmented region using the original colours of the ima
32      % labels found for k= 4
33 -    labels = repmat(pixel_labels, [1,1,3]);
34 -    clustered_peppers = peppers;
35
36 -    figure;
37 -    for k = 1:K
38 -        clustered_peppers = peppers(:,:,:) .* uint8(labels(:,:,1)== k);
39 -        subplot(2,2,k), imshow(clustered_peppers);
40 -        title(sprintf('Cluster %d', k));
41 -    end
42
```

Figure A.2.2: Code for Section 3 (continued)

```
1      %% SYDE 575 - LAB 5 - SECTION 4
2      %  Image Transform
3
4      %% 8x8 DCT Matrix
5 -    T = dctmtx(8);
6 -    f = double(rgb2gray(imread('lena.tiff')));
7
8 -    figure;
9 -    imshow(T);
10 -   title('8x8 DCT Transform Matrix');
11
12     %% Absolute Value of 8x8 DCT Sub-Images
13 -   F_trans = floor(blkproc(f-128,[8 8],'P1*x*P2',T,T'));
14
15 -   figure;
16 -   disp(F_trans(81:88,297:304))
17 -   imshow(abs(F_trans(81:88,297:304)),[]);
18 -   title('Abs Val at (81,297)');
19
20 -   figure;
21 -   imshow(abs(F_trans(1:8,1:8)),[]);
22 -   title('Abs Val at (1,1)');
23
24     %% Original 8x8 DCT Sub-Images
25 -   f = im2double(rgb2gray(imread('lena.tiff')));
26
27 -   figure;
28 -   imshow(f(81:88,297:304));
29 -   title('Original at (81,297)');
30
31 -   figure;
32 -   imshow(f(1:8,1:8));
33 -   title('Original at (1,1)');
34
```

Figure A.3.1: Code for Section 4

```
%% Reconstructed Lena
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;

figure;
subplot(2,1,1), imshow(f);
title('Original Lena');
subplot(2,1,2), imshow(f_thresh,[]);
title('Reconstructed Lena');

psnr = PSNR(f_thresh/256,f);
```

Figure A.3.2: Code for Section 4 (continued)

```
%% SYDE 575 - LAB 5 - SECTION 5
%  Quantization

%% Reconstruct image with 1z, 3z, 5z, and 10z
T = dctmtx(8);
f = double(rgb2gray(imread('lena.tiff')));
F_trans = floor(blkproc(f-128,[8 8],'P1*x*P2',T,T'));

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 x = [1,3,5,10]
    F_thresh = blkproc(F_trans,[8 8],'round(x./(P1))', x*Z);
    f_thresh = floor(blkproc(F_thresh,[8 8],'P1*(x.*(P3))*P2',T',T, x*Z))+128;

    figure;
    imshow(f_thresh,[]);
    title(sprintf('Reconstructed image with %dZ', x));

    format long;
    x, PSNR(f/256, f_thresh/256)
end
```

Figure A.4.1: Code for Section 5

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

        reluLayer

        fullyConnectedLayer(10)
        softmaxLayer
        classificationLayer];
cell_test = {Xtest ytest};
% Specify Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',100, ...
    '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);
```

Figure A.5.1: Section 6 Code - CNN

```
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)

```

Figure A.5.2: Section 6 Code - CNN (continued)

```
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
```

Figure A.5.3: Section 6 Code - MLP

```
% Specify Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',100, ...
    '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);

```

Figure A.5.4: Section 6 Code - MLP (continued)

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

Figure A.6.1: PSNR Code