**Lab 5: Image Compression and Segmentation**

****

Jonathan Tjong 20723414

William Li 20720929

**SYDE 575 - University of Waterloo**

Submitted To:Prof David Clausi

Due Date:Dec 7, 2021

# Introduction

In lab 5, our final lab of the course SYDE 575, a variety of concepts were discussed. Through MATLAB, we were able to experiment with the following fundamental image compression concepts: chroma subsampling, DCT transformations, and quantization. We also developed experience with colour segmentation and convolutional neural networks.

Firstly, we transformed the peppers image into the YCbCr colour space. We downsized the colour channels (Cb and Cr), as a form of compression, and rescaled them again to the original size (decompression). We did a similar transformation with the luma channel (Y channel), and compared the results of the two reconstructions.

We also experimented with colour segmentation in the L\*a\*b\* colour space, using k-means clustering. We observed the significance of using different values of k, and using different initial points.

Next, we studied the discrete cosine transform (DCT) using the Lena image. We used an 8×8 DCT transform matrix and examined the DCT of 2 sub-images from different locations in the image. We then experimented with discarding most of the DCT coefficients, similar to compression, and observed what the resulting reconstructed image looked like.

Expanding on this, we performed DCT compression using a quantization matrix, and ran multiple iterations of image reconstruction with various levels of quantization, we saw how the image changes as quantization level increases.

Lastly, we experimented with training a multi-layer perceptron (MLP) and a convolutional neural network (CNN) on a set of handwritten numbers. We observed how the performance changed for the two models as the following parameters were played with: number of epochs and learning rate.

# Chroma Subsampling

| **Original Peppers.png** |
| --- |

| **Y Channel** | **Cb Channel** | **Cr Channel** |
| --- | --- | --- |

**1. Describe the Cb and Cr channel images. Why do they appear this way?**

The chroma channels (Cb and Cr) have less image structure information and look like almost inverses of each other. In the Cb channel, yellows appear almost black and the purple background is lighter. In the Cr channel, red colours appear white. This is expected as Cb and Cr are colour differences channels from blue and red to luma respectively. This also makes sense as the purple background (between red and blue) looks identical in both Cb and Cr channels.

**2. Compare the level of image detail in the Cb and Cr images with the Y channel image. Which contains more fine details? What does that say about the luma (Y) and chroma (Cb and Cr) channels?**

The Y channel contains the most fine detail and image structure. This is because it only contains luminescence values and is essentially a grayscale image. This shows that the human eye is more sensitive to luminescence values and luminescence is more important for determining image structure. The Cb and Cr channels contain less relevant information for the human eye and on their own, they have less meaning and are harder to interpret.

| **Chroma sub-sampling** |
| --- |

**3. Compare the resulting image from chroma sub-sampling with the original image. How large are the visual differences?**

The differences are unnoticeable. The resampled chroma channels appear identical to the original image.

**4. Based on the resulting image, what can you say about chroma sub-sampling and its effect on image quality?**

Chroma sub-sampling is a very effective method of image compression that can easily and accurately be resampled in an identical manner. The effect on image quality is not noticeable or perceptible by humans. Since humans are less perceptive to colour information, it makes sense to compress the colour portions of images to save on bandwidth.

| **Luma sub-sampling** |
| --- |

**5. Compare the resulting image from luma sub-sampling with the original image. How large are the visual differences?**

This time, the luma upsampled channel creates a fuzzy looking image. The edges and overall structure loses the same edge crispness compared to the original image. The image has lost vital luminescence and structure information during the bilinear interpolation upscaling.

**6. Based on the resulting image, what can you say about luma sub-sampling and its effect on image quality?**

Luma sub-sampling is not an effective method of image compression as more visual information is lost in the luminescence channel. The image quality takes a noticeable hit as you can visually discern the loss of image quality in the luma sub-sampled image compared to the original.

**7. Compare the resulting image from luma sub-sampling with the image produced using chroma sub-sampling. Which method performs better? Which is better for reducing network bandwidth while preserving visual acuity? Why?**

Chroma sub-sampling is a much better method as the resulting image is identical to the original. Luma sub-sampling Chroma sub-sampling is also a more efficient method of compression and reducing network bandwidth, as two colour channels can be compressed rather than the single luma channel. Thus, chroma sub-sampling preserves image quality while also saving more network bandwidth compared to luma sub-sampling.

# Colour Segmentation

| **k=2** |
| --- |

| **Using the given start points for k = 4:** |
| --- |

**Random starting points for k=4**

|  |  |
| --- | --- |
|  |  |

**8. For the various values of k, how did the clustering change? Explain.**

The value of k indicates how many clusters should be sorted. By choosing k initial samples, each pixel is sorted into its minimum distance from each sample. As k changes, the number of unique clusters changes. In this case, minimum distance is calculated by using the euclidean distance in L\*a\*b\* space. The increased number of clusters means that the image should be divided into more of it’s base constituents, in this case, colours.

**9. What is the effect of the initial points on the final clusters? Does this impose any limitations? Why?**

The initial points can have an effect on where each cluster finds its minimum distance. Depending on the initial k centroid, the values may converge to a local minimum as opposed to finding the global minimums. This has the potential to skew other clusters into grouping pixels or colours that would otherwise find different global minimums. As you can see, in the 4 trials running k-means with 4 random clusters yielded different results. In some cases, the purple background was split into two clusters rather than finding the one global minimum. This imposes some limitations as you may need to run multiple trials of k-means clustering to find the true global minimum. One converging trial is not enough.

| **Cluster 1** | **Cluster 2** |
| --- | --- |
| **Cluster 3** | **Cluster 4** |

**10. Include an image of each cluster and comment on the segmentation performance.**

For this example of k=4, the k-means algorithm did a relatively good job of segmenting the colours into the appropriate clusters. You can see that the first cluster identified all of the purple background and separated it from the coloured peppers. It did also grab some of the white onion which must’ve been the closer in L\*a\*b\* space to purple than the other clusters. Cluster 2 managed to separate all of the yellows in the image as well as separating the lighter greens. Cluster 3 performed well in isolating all the oranges and red colours. Finally Cluster 4 separated all the dark greens as well as the darker whites on the onions.

# 

# Image Transform

| **8x8 DCT transform matrix** | **Plotting each row of the DCT transform matrix** |
| --- | --- |

**11. What does each row of the DCT transform matrix represent? Look at the pattern for each row. If you still don’t see it, try plotting each of the rows as a 1-D function**

Each row of the DCT transform matrix represents a 1D cosine function of varying frequency. Row 1 has 0 frequency, and row 2, row 3, etc have increasing frequency.

| **DCT of 8x8 subimage starting at (81, 297)** | **DCT of 8x8 subimage starting at (1, 1)** |
| --- | --- |

**12. Describe the energy distribution of the DCT of the sub-images. What does each pixel represent? Explain why DCT would be useful for image compression in the context of the DCT energy distribution.**

For both sub images, most of the energy is distributed in the low frequency discrete-cosine basis functions. In fact, in the second sub-image, all the energy is represented by the top left discrete-cosine basis function. In other words, the sub-image is a solid intensity and has no frequency.  
Each pixel in the DCT of the sub-images represents a DCT coefficient, which represents the amplitude of the discrete-cosine basis function component that is used to build up the sub-image. Each location of the pixel of the DCT corresponds to the coefficient of a different basis cosine. As we move down to the right and to the bottom, the frequency content of the basis cosine increases.  
The DCT is useful for image compression since it makes it very clear how an image can be built up with discrete-cosine basis functions, and gives a great idea to which frequency components are redundant and can be omitted from the image. In the second sub-image, there is only a DC coefficient, meaning that the image can be compressed by removing every single other discrete-cosine basis function. We only need one pixel to represent the image, since all the other basis cosines are redundant.

**13. Compare the DCT of the two sub-images. How are they different? Why? Explain in the context of the image characteristics at those locations and the DCT energy distribution.**

As mentioned earlier, both sub images have most of their energy distributed in the low frequency discrete-cosine basis functions. This is seen by the top left corner containing the brightest pixels of the DCT. The DCT of the sub-image starting at (1, 1) is different from the first DCT since it seems that only the very top left pixel is non-zero, and every other pixel is black. This indicates that the sub image has no frequency components and is a solid intensity, since only the DC discrete-cosine basis function is needed to represent the sub-image. On the other hand, the sub-image starting at (81, 297) is still mostly lower frequency, however still does contain frequency components.

| **Reconstructed image using only 6 DCT coefficients**    **PSNR = 29.8762** |
| --- |

**14. Describe how the reconstructed image looks compared to the original image. Why does it look this way?**

The reconstructed image looks blurred and unsharp compared to the original image.

All of the DCT coefficients were discarded from the DCT transform of the image, except for the 6 DCT coefficients in the most top-left. As a result, only the low frequency components in both directions (and the DC component) were preserved in the image. This is why the reconstructed image lacks sharpness and finer detail. Essentially the image was compressed and reconstructed in a lossy way, specifically losing the high frequency components.

**15. What artifact is most prominent in the image? Why does this artifact appear?**

The artifact most prominent in the image is block artifacts. This is observed in the image as visible boundaries between 8x8 sub-images. This artifact appears because each 8x8 sub-image is processed by the DCT independently and quantized/transformed independently without the context of the neighbouring sub-images, making it so that when the sub-images are reconstructed and composed together side by side, we see their boundaries.   
Also, the fact that all the high frequency components in each sub-image were lost as part of the process is a major reason for the blocking artifacts. In general, as quantization increases (in other words, more and more higher frequency components are discarded), the more that blocking artifacts become visually prominent. In our case, since only 6 DCT coefficients were preserved, this is considered high quantization.

**16. What conclusions can you draw about the DCT in terms of image compression? Does it work well? If yes, why does it work well?**

The DCT is a very effective tool for image compressions. This is because it allows us to see how an image can be built up of discrete cosine basis functions, and specifically which ones are important and which ones are redundant. Typically, the low frequency components are more important and are preserved as part of DCT-based compression, and high frequency components are more redundant and can be removed.

The human vision system is much more sensitive to variations in low frequency components compared to high frequency. This means that when high frequency components are removed as part of DCT quantization, there isn’t a significant noticeable loss in image quality.

# 

# 

# Quantization

**Reconstructed image with varying amounts of DCT Quantization/Compression**

| **Quantization Matrix = Z**    **PSNR = 35.6963** | **Quantization Matrix = 3Z**    **PSNR = 32.2807** |
| --- | --- |
| **Quantization Matrix = 5Z**    **PSNR = 30.3807** | **Quantization Matrix = 10Z**    **PSNR = 27.3817** |

**17. What happens to the DCT coefficients when quantization is performed? What effect does it have on image quality?**When quantization is performed, DCT coefficients are divided by the quantization matrix and rounded. This ends up with many DCT coefficients becoming 0. This results in certain discrete cosine basis functions being removed from the image (usually the higher frequency cosines), resulting in slightly reduced image quality. This is a lossy transformation (once a coefficient becomes zero, we can’t back the original coefficient) and we lose detail in the image.

**18. Compare the reconstructed image produced using 3Z with the original image. Why does the reconstructed image look this way?**

The reconstructed image produced using 3Z looks similar to the original image, however has slightly poorer image quality. The image is slightly less sharp, and also some blocking artifacts can be seen. This is a result of the quantization performed during the reconstruction. Some information is lost as part of the quantization, so it's impossible for the image to look like the original.

**19. Compare the reconstructed images produced by the different levels of quantization, as well as the PSNR for each reconstructed image. What happens as the level of quantization increases?**

As the level of quantization increases, i.e. we use larger multiples of Z, the reconstructed image becomes poorer and poorer in quality. This is because as quantization increases, more and more DCT coefficients become zero and as a result the image is more and more compressed. Less and less of the higher frequency information is preserved as quantization increases, so the resulting images look more blurry and poorer quality. As well, we can see that blocking artifacts become more and more visible as quantization increases.

**20. Which artifact becomes more prominent as the level of quantization increases? Why?**

Again, blocking artifacts become more prominent as the level of quantization increases. This is because more and more of the higher frequency information is discarded as the level of quantization increases, which correlates to more visible blocking artifacts.

**21. What conclusions can you draw about the quantization process? Explain in the context of the trade-off between compression performance and image quality.**

In conclusion, quantization is a helpful tool for compressing images and reducing the size needed to them. As we increase the level of quantization, more and more DCT coefficients are discarded, resulting in a higher degree of compression, since less coefficients are needed to represent the image. However, as seen in this section, this has a trade-off with image quality. As the quantization level increases, image quality reduces and more artifacts are seen in the reconstructed image. Therefore there is a fine balance to applying quantization during image compression, and there is some thought that must go into the quantization process (how much should the image be quantized).

# Convolutional Neural Networks

**22. In terms of number of parameters and runtime, which type of layer is more expensive? A fully connected layer or a convolutional layer? Why might a convolutional layer be beneficial for images?**

A fully connected layer uses much more parameters/weights than a convolutional layer, since each output node has a weight correlated to each input node (uses matrix multiplication). On the other hand, a convolutional layer uses a spatial filter which operates on local neighborhoods, meaning that much less parameters/weights are required.  
For runtime, convolutional layers are slower due to the nature of their operation (sliding window that moves, many computations for a convolutional layer), vs. a single matrix multiplication which needs to be performed for a fully connected layer.

A convolutional layer is beneficial for images since they take advantage of the spatial correlation that exists in natural images. This is not the case for fully connected layers.

| MLP [Epoch = 10, Learning rate = 0.01]  test\_accuracy = 0.9049  train\_accuracy = 0.9464 |
| --- |

| CNN [Epoch = 10, Learning rate = 0.01]  test\_accuracy = 0.9511  train\_accuracy = 0.9726 |
| --- |

**23. What can you observe about the training graphs? Does one train quicker? Does one converge to a lower error rate, training score, or testing score? Have they converged?**

In both training graphs, the models seem to converge. Both graphs have approximately flatlined at above 90% accuracy. We can observe that the majority of accuracy gains are achieved in the first few epochs, and the last few percentage gains are achieved over many more epochs.  
MLP trains much quicker compared to CNN, with an 8s elapsed training time compared to 21s elapsed training time. However, the CNN makes up for this with better testing accuracy and better training accuracy, after training is complete.

**24. Is the testing accuracy similar to the training accuracy?**

The testing accuracy is similar to training accuracy, however it is not exact. In both models the test accuracy is slightly lower.

MLP: test accuracy is 4.15% lower than training accuracy

CNN: test accuracy is 2.15% lower than training accuracy

This is potentially a sign of minor overfitting in the model.

| MLP[Epoch = 10, Learning rate = 0.001]  test\_accuracy = 0.8062  train\_accuracy = 0.8162 |
| --- |

| CNN [Epoch = 10, Learning rate = 0.001]  test\_accuracy = 0.8283  train\_accuracy = 0.8366 |
| --- |

| MLP[Epoch = 10, Learning rate = 0.1]  test\_accuracy = 0.9382  train\_accuracy = 0.9998 |
| --- |

| CNN [Epoch = 10, Learning rate = 0.1]  test\_accuracy = 0.9673  train\_accuracy = 0.9913 |
| --- |

**25. Try raising and lowering the learning rate by factors of 10. Comment on both the speed of convergence and the stability of the training graph as you change the learning rate**

Decreasing the learning rate to 0.001 slowed the training such that progress got stuck and the loss function never converged. This proved to be too slow of a training rate for the number of epochs. On the other hand, having a learning rate too high can cause rapid changes and may cause the model to converge on a suboptimal solution. This didn’t seem to occur at a learning rate of 0.1 as the model still managed to converge to an accurate model resulting in ~95% testing accuracy on both models. The 0.1 training rate reached convergence within only a few epochs whereas the 0.001 training rate did not manage to converge within 10 epochs. For these models and training sets, a learning rate of 0.1 seemed to be the optimal learning rate.

| MLP[Epoch = 100, Learning rate = .01]  test\_accuracy = 0.9301  train\_accuracy = 1 |
| --- |

| CNN[Epoch = 100, Learning rate = .01]  test\_accuracy = 0.9698  train\_accuracy = 1 |
| --- |

**26. For both networks, determine if overfitting occurred for 10 epochs and 100 epochs using the original learning rate of 0.01. Discuss factors you think could have contributed to your outcome.**

During both network’s training for 10 and 100 epochs, overfitting did not occur. There is not a significant difference between testing and training accuracy that would indicate overfitting. Although both models reached 100% training accuracy (which can be an indicator of overfitting), the testing accuracies still remained quite high, showing that the model is indeed not overfitted. Some significant factors that contributed to no overfitting could be the large amount of training data. There are 5000 samples of training images of numbers 0-9. This large amount of data helps the model generalize the entire population such that the testing accuracy is still high.

# 

# Conclusion

In conclusion, throughout this lab several important deductions were made in the areas of chroma sub-sampling, colour segmentation, image transform, quantization and convolutional neural networks.

When we transformed the images into the YCbCr colour space, we performed chroma-subsampling and luma sub-sampling to compare the results. It was determined that chroma-subsampling retained greater visual acuity and was a more efficient model for compressing data. This is due to how the human eye operates, as it is much more sensitive to luminescence data compared to colour data.

During our experimentations with colour segmentation, we observed how changing the number of clusters in a k-means algorithm affected what local and global minimums the image converged to. We also saw how random k-centroids can affect the end result compared to setting initial points.

For discrete cosine transform (DCT) it was concluded that DCTs were a very effective method of compression with minimal visual acuity loss. Discarding all but 6 DCT coefficients still produced an image with a relatively high PSNR value of 29.8. Using a quantization matrix also yielded positive results. The quantization matrix discards a subset of high frequency DCT coefficients resulting in image compression while retaining high visual acuity. Using the quantization matrix Z, a PSNR value of 35 was retained. We also noticed the degrading effects of using higher n\*Z quantization matrices and how they introduced blocking artifacts.

Lastly, during the machine learning experimentation, we saw how a low learning rate could fail to converge and how a higher learning rate increases rate of convergence at a trade-off of potential convergence to a suboptimal solution. We also saw first hand how a large training database can prevent overfitting despite a large training period.

# 

# 

# Appendix

**Part 2 code:**

close all;

RGB = imread('peppers.png');

figure

imshow(RGB);

YCBCR = rgb2ycbcr(RGB);

figure;

subplot(1,3,1)

Y = YCBCR(:,:,1);

imshow(Y)

title('Y component');

subplot(1,3,2)

Cb = YCBCR(:,:,2);

imshow(Cb)

title('Cb component');

subplot(1,3,3)

Cr = YCBCR(:,:,3);

imshow(Cr)

title('Cr component');

CbUpscaled = imresize(imresize(Cb, 0.5), 2, "bilinear");

CrUpscaled = imresize(imresize(Cr, 0.5), 2, "bilinear");

YCBCR(:,:,2) = CbUpscaled;

YCBCR(:,:,3) = CrUpscaled;

chromaRestoredRGB = ycbcr2rgb(YCBCR);

figure

imshow(chromaRestoredRGB);

YUpscaled = imresize(imresize(Y, 0.5), 2, "bilinear");

YCBCR(:,:,1) = YUpscaled;

YCBCR(:,:,2) = Cb;

YCBCR(:,:,3) = Cr;

lumaRestoredRGB = ycbcr2rgb(YCBCR);

figure

imshow(lumaRestoredRGB);

**Part 3 code:**

close all;

RGB = imread('peppers.png');

% figure

% imshow(RGB);

C = makecform('srgb2lab');

lab = applycform(RGB,C);

ab = double(lab(:,:,2:3)); % NOT im2double

m = size(ab,1);

n = size(ab,2);

ab = reshape(ab,m\*n,2);

% 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(lab,1) size(lab,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');

colormap('jet')

%pixel\_labels(pixel\_labels~=1) = 0;

% pixel\_labels(pixel\_labels~=2) = 0;

% pixel\_labels = pixel\_labels./2;

% pixel\_labels(pixel\_labels~=3) = 0;

% pixel\_labels = pixel\_labels./3;

pixel\_labels(pixel\_labels~=4) = 0;

pixel\_labels = pixel\_labels./4;

imshow(RGB .\* uint8(pixel\_labels));

**Part 4 code:**

close all;

f = double(rgb2gray(imread('lena.tiff')));

figure

imshow(f, [])

T = dctmtx(8);

figure

imshow(T);

figure

plot(T(1,:));

hold on

plot(T(2,:));

plot(T(3,:));

plot(T(4,:));

plot(T(5,:));

plot(T(6,:));

plot(T(7,:));

plot(T(8,:));

hold off

legend("row 1", "row 2", "row 3", "row 4", "row 5", "row 6", "row 7", "row 8")

F\_trans = floor(blkproc(f-128,[8 8],'P1\*x\*P2', T, T'));

% figure

% imshow(abs(F\_trans), []);

% figure

% imshow(f(81:88, 297:304));

figure

imshow(abs(F\_trans(81:88, 297:304)), []);

figure

imshow(abs(F\_trans(1:8, 1:8)), []);

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;

%imshow(mask, [])

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

imshow(f\_thresh, [])

psnrLena = psnr(uint8(f\_thresh), uint8(f));

**Part 5 code:**

close all;

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];

Z = 10.\*Z;

f = double(rgb2gray(imread('lena.tiff')));

T = dctmtx(8);

F\_trans = floor(blkproc(f-128,[8 8],'P1\*x\*P2', T, T'));

F\_Quantized = round(blkproc(F\_trans, [8 8], 'x./P1', Z));

f\_reconstructed = floor(blkproc(F\_Quantized, [8 8], 'P1\*(x.\*P3)\*P2', T', T, Z)) + 128;

figure

imshow(f\_reconstructed, [])

psnrLena = psnr(uint8(f\_reconstructed), uint8(f));