SD 575 Image Processing

Fall 2023

Lab 5: Image Compression and Segmentation
Due **Monday** December 11th, 2023 at 11:59pm

Note: All lab reports will be submitted on Learn to group dropboxes.

Note: The lab report will be submitted on Learn to group drop boxes.
For help on how to use the Python functions mentioned throughout the lab, please use the links:

https://numpy.org/doc/stable/
https://scikit-image.org/
https://docs.scipy.org/doc/scipy/

1 Overview

The goal of this lab is to provide some hands-on experience with fundamental image compression concepts and techniques. Due to the exponential growth in usage and storage of digital graphic media, image compression has been very important in helping reduce storage and transmission bandwidth problems. Many real-world applications depend heavily on image compression, such s digital photography, video games, digital movie archival/streaming, and medical imaging. For this lab, we will study some fundamental image compression techniques such as chroma subsampling, image transform, and quantization.

The following images will be used for testing purposes:

- lena.tif
- peppers.png

These images can be found on the course website.

2 Chroma Subsampling

First, we will study the effects of chroma subsampling on image quality and how it can be used to provide image compression. For this study, we will use the peppers image. First, let us convert the image from the

RGB colorspace into the YCbCr colorspace using the `rgb2ycbcr` function from `skimage.color`. Plot each of the image channels (Y, Cb, and Cr) separately.

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

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. The `resize` function from `skimage.transform` will come in handy. Also, you will need to separate the color image into three separate images so you can operate on them independently. Recombine the original Y channel image and the two upsampled Cb and Cr images to create a new color image. This can be considered a simple way of reducing network bandwidth for image/video transfer – that is, downscaling the chroma images on the server, sending them over a network to a client, and rescaling on the client.

3. Compare the resulting image from chroma sub-sampling with the original image. How large are the visual differences?
4. Based on the resulting image, what can you say about chroma sub-sampling and its effect on image quality?

Let us study the effects of luma sub-sampling. Reduce the resolution of the luma (Y) channel by a factor of 2 in both the horizontal and vertical directions and then upsample it back to the original resolution using bilinear interpolation. Recombine the upsampled Y channel image and the original Cb and Cr images to create a new color image.

5. Compare the resulting image from luma sub-sampling with the original image. How large are the visual differences?
6. Based on the resulting image, what can you say about luma sub-sampling and its effect on image quality?
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?

3 Colour Segmentation

Image segmentation is useful in identifying regions of interest in an image to gain a more meaningful representation of the image. In this section, colour segmentation is explored using k-means classification to segment various colours in an image using the RGB and $L^*a^*b^*$ colour spaces. The $L^*a^*b^*$ colour space

consists of a luminosity dimension (L^*) and two colour dimensions called a^* and b^*. The a^* channel indicates colour which falls along the red-green axis while the b^* channel indicates colours which fall along the blue-yellow axis.

Load the `peppers.png` image in Matlab and convert the image to the $L^*a^*b^*$ colour space. (Hint: `rgb2lab` from `skimage.color` is useful here.)

Then, classify the colours in the $L^*a^*b^*$ colour space using k-means clustering for $k = 2$ and $k = 4$. Use `KMeans` from `sklearn.cluster` to do colour segmentation with the following starting point matrix, μ.

```
#K = 2
#row = np.array([55, 200]) - 1
#col = np.array([155, 400]) - 1

K = 4
row = np.array([55, 130, 200, 280]) - 1
col = np.array([155, 110, 400, 470]) - 1

mu = p_lab[row,col]
```

Then, reshape the $L^*a^*b^*$ channels:

```
m, n, ch = p_lab.shape
p_lab = np.reshape(p_lab, (m * n, ch), order='F')
```

Output the cluster indices and show the resulting classification for a given k, using the following code, where `cluster_idx` are the cluster indices. This should show in one image, each cluster with its own unique colour.

```
% Label each pixel according to k-means
pixel_labels = np.reshape(cluster_idx, (m, n), order='F');
plt.imshow(pixel_labels, cmap='jet')
plt.title('Image labeled by cluster index');
```

8. For the various values of k, how did the clustering change? Explain.
9. What is the effect of the initial points on the final clusters? Does this impose any limitations? Why?

Next, output each cluster/segmented region using the original colours of the image using the `pixel_labels` found for $k = 4$. When showing the segmented regions for a particular pixel label, other pixels should be set to zero.)

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

4 Image Transform

Let us now study the discrete cosine transform (DCT) and the characteristics of an image in the transform domain. The DCT decomposes an image into a series of sinusoids with different amplitudes and frequencies. In block transform coding algorithms, the image is divided into smaller sub-images and each sub-image is transformed using an image transform separately. Perhaps the most popular image transform for block transform coding is the DCT. One efficient method for computing the DCT of a sub-image is to use the DCT transform matrix. The DCT transform matrix can be constructed using `dctmtx` function from the `lab5template.py` file. For this study, we will use the 8×8 DCT transform matrix `T` and use grayscale Lena image `f` as the test image. Plot the 8×8 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.

Now apply the DCT transformation matrix on each 8×8 sub-image. This can be performed as follows:

`F_trans = np.floor(blockproc(f-128, T, [8, 8], func))` [1] where f $\in [0, 255]$. This returns an image where each 8 × 8 patch is the DCT applied to the corresponding 8 × 8 patch of lena. Plot the **absolute value** of the DCT of the 8×8 sub-image with top-left corner at $(row, col) = (80, 296)$ and the DCT of the sub-image with top-left corner at $(row, col) = (0, 0)$.

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.
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.

Now let's try discarding all but 6 of the DCT coefficients in each sub-image and then reconstructing the image. This can be done by first applying a threshold to the sub-images,

```
mask = np.zeros((8,8))
mask[0,0] = 1
mask[0,1] = 1
mask[0,2] = 1
mask[1,0] = 1
mask[1,1] = 1
mask[2,0] = 1

F_thresh = blockproc(F_trans, mask, [8, 8], func1)
# see lab5template.py file for func1
```

[1] The `func` function (see `lab5template.py`) calculates QXQ^T in matrix algebra terms, where Q is the DCT matrix and X is an 8 × 8 sub-image extracted from the original image.

and then performing an inverse DCT on the sub-images

```
f_thresh = np.floor(blockproc(F_thresh, T.T, [8, 8], func)) + 128
```

Plot the reconstructed image and the corresponding PSNR.

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

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

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?

5 Quantization

One of the most important steps in lossy image compression stage is the quantization step. It is highly desired that the transform coefficients of a sub-image is quantized in such a way that the amount of data needed to represent the image is greatly reduced without causing undesirable artifacts. Let us now study the effects of different levels of quantization on image quality. For this study, the grayscale Lena image will be used as the test image. First, we will construct the quantization matrix used in the JPEG standard:

```
Z = np.array([
              [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]
            ])
```

Now perform the 8×8 DCT transform on the Lena sub-images (remember to subtract 128). To perform quantization on the sub-images, divide the sub-images (as floating point) by `Z`, and then round the resulting quantized DCT to integers. To reconstruct the image, multiply the quantized DCT sub-images by `Z` and then perform the inverse DCT transform on the sub-images (remember to add 128). Plot the resconstructed image and the corresponding PSNR. Now perform the above quantization process again on the image, but this time using `3Z`, `5Z`, and `10Z`. Plot the reconstructed images and the corresponding PSNR.

17. What happens to the DCT coefficients when quantization is performed? What effect does it have on image quality?

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

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?

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

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

6 Convolutional Neural Networks

Neural networks are a powerful class of prediction models that have many applications, such as pattern recognition tasks like image classification. In this section we will explore the usage of neural networks to classify images of written digital characters from the MNIST dataset (posted on Learn).

We will compare a multi-layer perceptron (MLP) and a convolutional neural network (CNN). The networks are composed of two types of layers: **fully connected layers** or **convolutional layers**.

A fully connected layer collapses the previous layer into a vector, and multiplies it with a weight matrix, such that each node in the current layer uses a linear combination of each element in the previous layer. Each fully connected layer applies a non-linear function to the result of this multiplication.

$$\mathbf{h}^t = \mathrm{f}(W^t \mathbf{h}^{t-1} + \mathbf{b}^t) \tag{1}$$

for layer t, and trainable matrix W^t and bias vector $\mathbf{b}^t$, and non-linear function f

A convolutional layer represents an image array with dimensions of width, height, and channel. Each channel in a convolutional layer applies a non-linear function to the summation of each channel in the input convolved with a trainable filter (usually 3×3 or 5×5).

$$H^t[..,..,o] = f(\sum_i W^t[..,..,i,o] * H^{t-1}[..,..,i] + B^t) \tag{2}$$

which indicates that the o^{th} channel of layer H^t applies a non-linear function to the sum of responses of convolutions between filters W^t and each channel (i) of the previous layer H^{t-1}.

To train the networks, we will be using a gradient descent algorithm. This takes a loss function, L, and computes the partial derivatives of the loss with respect to each parameter for a given mutually exclusive subset, or "batch", of the training set. Thus, each parameter w is updated with

$$w \iff w - \alpha \frac{\partial L}{\partial w} \tag{3}$$

with a learning rate α. Applying this for all batches in the training set is called an "epoch". Generally we will train the model on several consecutive epochs until the loss function is at an acceptable level for the training set.

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?

The code to construct and train two neural networks on MNIST data is posted on learn as `mlp.m` and `CNN.m`. All you need to do is tune the epochs and learning rate parameters.

Try training the MLP and the CNN on the training set for 10 epochs with a learning rate of 0.01 and plot the training graphs of accuracy vs epoch. Also, record the accuracy of each model on the testing set after training.

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 have converged?
24. Is the testing accuracy similar to the training accuracy?
25. Try raising and lowering the learning rate by factors of 10. Comment on both the speed of convergance and the stability of the training graph as you change the learning rate.

Due to their high number of trainable parameters, neural networks have a high capacity for adapting to complex relationships. Some dangers of this is that a network with more parameters can overfit to the training set and not generalize well to the testing set (i.e. there is a large discrepencacy between the training accuracy and testing accuracy).

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.

7 Report

Include in your report:

- A brief introduction.
- Pertinent graphs and images (properly labelled).
- Include code (can be included in appendix).
- Include responses to all questions. Enumerate your answers to each question (e.g., 1., 2. or Q1, Q2, etc.).
- A brief summary of your results with conclusions.