ECE 253, Homework 6

Due: Wednesday November 30, 2022 by 11:59pm

The first three problems should be done by hand, and the last one involves Matlab. Submit your homework electronically in Canvas on Gradescope. Everything can be uploaded as one PDF file– include your answers to each question and your Matlab code (cut and paste it in). Include your full name and PID.

Note: This homework and its solution set are meant as study aids rather than for assessment, and so the homework will not be graded for correctness; the TAs will just check that each problem was attempted and will assign full credit if it was attempted, whether right or wrong. The solution set will be released over the weekend before the homework is due, and you are welcome to check your approach or your answers against the solution set prior to turning it in.

1) **Truncated Huffman Coding**

Consider the alphabet with 14 possible symbols and their associated probabilities given on slide 22 of Pre-Lecture19.

(a) Construct a full Huffman code for this source. Show your source reduction steps, and give the final codewords for each symbol.

(b) What is the entropy of this source?

(c) What is the expected length and the efficiency of your full Huffman code? How do they compare to the expected length and the efficiency of the truncated Huffman code for this source given on slide 23?

2) **Arithmetic Coding**

Consider a four-symbol source $\{a, b, c, d\}$ with source probabilities $\{0.1, 0.4, 0.3, 0.2\}$. Arithmetically encode the sequence bba. Use the division of the unit interval where symbol a corresponds to the sub-interval [0, 0.1), symbol b corresponds to the sub-interval [0.1, 0.5), etc. Use the interval rescaling approach. For each input symbol, show the corresponding interval(s) and the bits put out, if any.

3) **DCT and DFT**

Let $f(k)$ denote a one-dimensional N-point sequence that is zero outside $0 \leq k \leq N-1$. The unitary N-point DCT is given by

$$C(u) = \alpha(u) \sum_{k=0}^{N-1} f(k) \cos\left[\frac{(2k+1)u\pi}{2N}\right]$$

where

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{if } u = 0 \\ \sqrt{2/N} & \text{if } u = 1, 2, \ldots, N-1 \end{cases}$$

Let $g(k)$ be the reflected sequence:

$$g(k) = \begin{cases} f(k) & \text{for } k = 0, 1, \ldots, N-1 \\ f(2N-1-k) & \text{for } k = N, N+1, \ldots, 2N \end{cases}$$

The 2N-point unitary DFT of $g(k)$ is:

$$G(u) = \frac{1}{\sqrt{2N}} \sum_{k=0}^{2N-1} g(k) e^{-j2\pi uk/2N}$$

Derive the relationship stated in class between the 1D N-point DCT of $f(k)$ and the 1D $2N$-point DFT of $g(k)$.

4) **JPEG**

a) Read in the image totem-poles.tif. How many bits per pixel is this initial representation of the image? To answer this, simply find the number of pixels in the image, find the number of bytes that the file takes up on your computer's storage, and calculate bits per pixel (bpp).

b) Generate a curve of of PSNR versus rate (bits per pixel) for compressing this image with baseline sequential JPEG. You can do this by using imwrite to write out the image as a JPEG file with different quality factors. Each time you write it out, you can check the size of the output file, and read the image back in to find the PSNR, for example:

```
>> imwrite(totem,'totem50.jpg','Quality',50)
>> totem50 = imread('totem50.jpg');
>> impsnr50 = psnr(totem,totem50,255)
impsnr50 =
34.7750
```

Try quality factors of 1 (worst quality), 70 (very high quality), and at least 4 points in between. Create a table that has columns for quality factor, bits per pixel, and PSNR. Also plot your results as PSNR versus bpp. Look at the images and the PSNR curve and discuss the results in terms of what artifacts dominate at low bit rates, at what rate is the image visually the same as the original, and the general shape of the curve (e.g., does PSNR increase linearly with bit rate?).

c) Let's see how downsampling can play into low rate coding. Use imresize to downsample the totem image by a factor of F in each direction, where you are asked to try both F=2 and F=4. Compress each small version at different quality factors as in the previous part. When you read the image back in, use imresize to upsample the image back to the original size, and compute the PSNR relative to the original full size totem image. Plot the original PSNR vs. bpp curve and the ones for F=2 and F=4 on the same plot (so bpp is always referring to the bits per pixel at the original full size). Choose your set of quality factors such that you cover both the very low rates which are below what can be achieved by full-size JPEG encoding, and you also overlap somewhat the set of rates achieved by full-size coding, so you can compare directly with it. Discuss your results.

d) Lastly, consider the image "totem1" which comes from compressing the original full size totem image using a quality factor of 1. If you are trying to be an extra smart JPEG decoder, what can you do to improve this received image? Try at least one improvement and report your PSNR relative to the original totem image.