ECE-178 Homework #6

Due on Thursday, November 21, 2024, 11:59 PM

Name:

Uncompressed Baseline

Using PIL, load in `nasa.jpg` and convert it to a numpy array (keep all three color channels). Save this numpy array using `numpy.save('uncompressed.npy', uncompressed, allow_pickle=False)`. How large is your uncompressed image?

Uniform Scalar Quantization

1. How many bits do you need to represent each value in each channel?
2. Let's say that we can only afford 4 bits per pixel. Derive a transformation which reduces each value in your matrix to a 4 bit value and an inverse transformation which converts a 4 bit value back into your answer for (1).
3. What is the worst possible quantization error under this scheme? In other words, find the maximum value of $e = |T^{-1}(T(x)) - x|$, where $T(x)$ is your transformation, $T^{-1}(x)$ is your inverse transformation, and x is the value of one of your channels in your image.
4. Implement your transformation for each pixel in the original image. You may find the provided `twopack` function useful to encode your 4-bit values (combining two 4-bit values into one 8-bit value). Save the transformed image using `numpy.save` as `uniformly_quantized.npy`. *Warning: `twopack` only supports packing an even number of values*
5. Implement and apply the inverse transformation to your quantized numpy array. This is your unquantized image.
6. Report the distortion and compression ratio. Does this make sense?

   a. Distortion is defined as the average squared error between the original image and the unquantized image from (5).

   b. Compression ratio is the size of `uncompressed.npy` divided by the size of `uniformly_quantized.npy`.

Linear Estimator

1. Suppose the original image is sampled such that only every other pixel is saved in full quality. Derive an expression for a 1st order linear estimator which estimates each unknown pixel using the value of one neighboring pixel.
2. Implement the sampling method described in (1) in Python. Then, using your estimator in (1), attempt to reconstruct the values which are missing from the sampled array. Compute an error matrix which contains the error between your reconstruction of the sampled image and your original image. Then save your sampled image and the error matrix as `.npy` files.
3. Using the error matrix and the sampled image, how would you reconstruct the original image with zero distortion? Describe your method mathematically. Then, implement the reconstruction function in Python.
4. Report the distortion and compression ratio. Explain why the result makes sense.

Predictive Compression

1. Use a 4-bit uniform scalar quantizer to quantize the error matrix values (leave the sampled values unchanged). How does this change the distortion and compression ratio?
2. Plot a histogram of the error matrix values (across all channels). Based on this, propose a new 16-level uniform quantizer which would improve your distortion without reducing your compression ratio.
3. Implement compression using the improved uniform quantizer. Report the distortion and compression ratio, and explain why this makes sense.
4. Without modifying the quantizer, how would you improve the compression ratio?
5. Optimize your compressor/decompressor and report your “best” distortion and compression ratio pair. You must use a uniform scalar quantizer, an nth order linear estimator and a sampling technique, but the details are up to you.