**Wavelet Transform**

Wavelet transforms are used to compress images, and as such, provide a good mechanism to reduce the complexity of an image, and to thereby make it more amenable to machine. Another advantage of their compression algorithm is that it’s kind of a telescoping algorithm. It makes a rough image, and then a rougher one, and then a rougher one, etc., all the way until it’s approximated the image as a single pixel. And so we can choose how detailed a picture we want depending on how accurate an image we need for our machine learning purposes. So for example, consider the following 8×8 black and white image (actually 1’s would be white and 0’s black – whatever).

A grid of numbers and equations

Description automatically generated

Now we’ll perform a wavelet-transform on our image. The transform consists of passing four 2×2 pixel filters, of sorts, across the image. These filters are defined below:

A group of math equations

Description automatically generated

The LL filter basically picks up the average pixel value. The other three filters pick of differences. The LH filter is sensitive to horizontal differences in pixel values. The HL filter is sensitive to vertical differences in pixel values. And the HH filter is sensitive to diagonal differences. Passing each filter over our image will result in a separate reduced 4×4 matrix. We can collect these four filtered matrices together and put them into the wavelet transform super matrix ([LL, LH], [HL, HH]). If we do that with our image above, we’ll get:

A colorful squares with numbers and symbols

Description automatically generated

As can see, we have just as much information (64 pixels) in the wavelet transform matrix as we did in the prior image matrix (64 pixels). Whereas in the image matrix, the pixel values of each 2×2 subblock were given, in our wavelet transform matrix, we have specified the average, horizontal difference, vertical difference, and diagonal difference of each 2×2 subblock. So same number of parameters in each case. The approximation to our image consists of extracting out the LL1 block from the wavelet transform. Can see that if we color in the non-zero pixel values, we get the best approximation to our original image we can get in a 4×4 matrix.

A grid of numbers and symbols

Description automatically generated

But we don’t have to stop here. We can proceed with a wavelet transform of this LL1 block. Applying the same filters, and grouping them together in the same fashion, we’d find:

A colorful squares with numbers and symbols

Description automatically generated

Again, we have the same number of parameters in our wavelet transform of the LL1 block as we do in the original LL1 block itself. The wavelet transform just reparameterized the information into average, horizontal difference, vertical difference, and diagonal difference values. The LL2 subblock corresponds to the average (×2) pixel values, and gives us a representation of the image. If we color in the non-zero values, we get:

A square with black text

Description automatically generated with medium confidence

which is, I guess, the best representation of our image in a 2×2 matrix. And we can repeat the process one more time. Taking the wavelet transform of our LL2 block, we come to:

A math equation with numbers and symbols

Description automatically generated with medium confidence

Repeating myself ad nauseum, we have the same information content in LL2 as we do in WT(LL2). But if we extract the LL3 submatrix, this will consist of the best approximation to our image in a 1 pixel block. We have:

A black and white symbol

Description automatically generated

Note we can equivalently express the entire original image matrix with the following matrices:



These telescoped matrices are often displayed as:

A table of equations with numbers and symbols

Description automatically generated with medium confidence

So you can see that at each level/pass, the wavelet transform preserves the information in the original image. Anyway, so for ML purposes, we might be able to replace an image with some [LLm, (LHm, HLm, HHm)], or maybe some subset of the four, or maybe even just LLm itself. And this would probably greatly reduce the computational cost.