lossy image compression
last time: lossless image compression (jpeg)
block transform coding
input image N*N — break into n*n blocks — forward n*n transform each block — quantizer (lossy) — symbol encoder — compressed image

eg. Huffman

decompress:
compressed image —> symbol decoder
inverse n*n transform —> reassemble blocks
—> decompressed image
n = 8, 16 discrete cosine transform

what 2D transform to use?
-DFT :(complex
- DCT (jpeg) real valued, compact energy into low frequency
- Hadamand - Haar - wavelet (jpeg 2000)

$$T(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} I(x,y)c(x,u)c(y,v) \qquad u,v \in [0, n-1]$$

$$c(x,u) = \begin{cases} \frac{1}{\sqrt{n}} & u = 0 \\ \sqrt{\frac{2}{n}} \cos\frac{(2x+1)u\pi}{2n} & u = 1, \dots n-1 \end{cases}$$

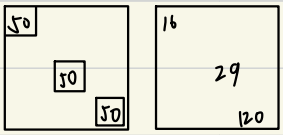
8*8 original image pixels -> 8*8 DCT coefficients

possibility for compression:
- zonal coding: how many bits per coefficient as a function of (u,v)
- threshold coding
  1) select the m largest coefficients, throw others away
  2) select the coefficients that account for (say) 95% of the total energy in the block
  3) select all coefficients above a threshold τ

How jpeg activity works:
specify a normalization matrix for T(u,v). This implicitiy specifies how many “levels” for each coefficients

$\hat{T}(u,v) = \text{round}\left(\frac{T(u,v)}{N(u,v)}\right)$ upper left: $\text{round}\left(\frac{50}{16}\right) = 3$

3*16 = 48



middle: $\text{round}(\frac{50}{29}) = 2$

lower right: $\text{round}(\frac{50}{120}) = 0$

reconstruction: $\dot{T}(u,v) = \hat{T}(u,v)N(u,v)$

2*29 = 58, 0*120 = 0

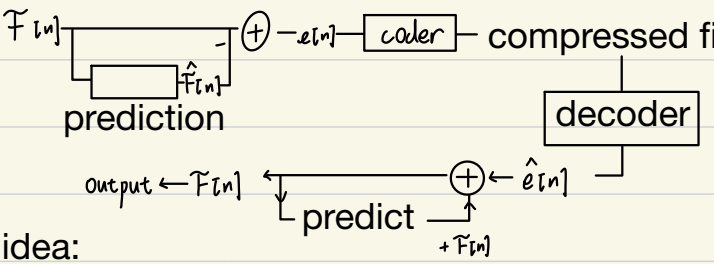
the smaller the value in N(u,v), the more accurate the reconstruction coefficient

To adjust jpeg quality, the N(u,v) can be multiplied by a # less than 1 => higher quality, or # greater than 1 => worse quality

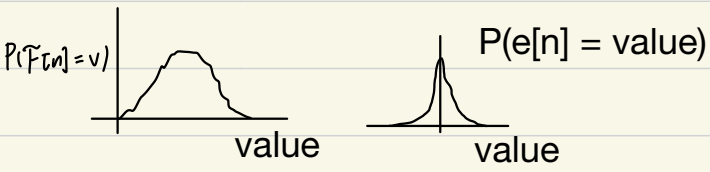
further details:
after quantization, jpeg re-orders the coefficients in a zig-zag pattern, then applies a lossless coder (eg. Huffman, RLE)

DC coefficient is coded separately w.r.t the DC for the previous sub-block
for color, convert to a luminance/ chrominance color space. Intensity + 2 color channels. Chroma channels coded at lower bitrate.

A few words on predictive coding:
idea: images have a lot of correlation between neighbouring pixels, so use previous pixels to predict the value of the next pixel, code the error in prediction.



idea:



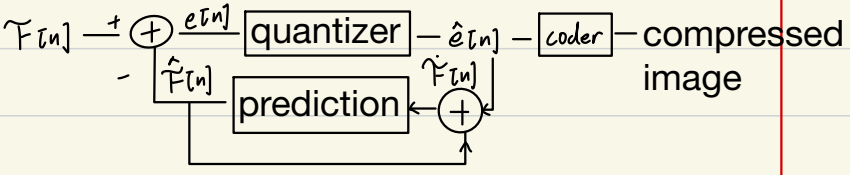
eg. $\hat{F}(x,y) = \text{round}\left(\sum_{i=1}^{m} a_i F(x,y-i)\right)$

weighted avg of previous pixels in row

$= \text{round}\left(\sum_{(u,v)\in \text{neigh of } (x,y)} a_i F(u,v)\right)$

aka autoregressive progess

if lossless coding of e[n], no problem.
if lossy coding of e[n], then we also need the quantizer inside the predictor to make sure the compressor/decompressor agree on predictions



Video compression
images are spaced very closely
many/most video compression algos use block-based motion compensation