Compression-based Forensics I

---CS355: Digital Forensics---

Dr. Yu Guan,
Department of Computer Science
University of Warwick
Yu.Guan@warwick.ac.uk

- **Image Fundamentals**
  - **understanding how a digital image is formed and stored**
- Image Enhancement
  - understanding how to remove artifacts from image and enhance it for forensic applications
- Forgery Detection
  - to be able to answer if an image is authentic, to localize which part of the image has been tampered
- Source Identification
  - to be able to identify the imaging source (device)

Image Fundamentals

Week 1
Image Acquisition
-Sampling
-Quantization

Week 2
Image Color Space
- RGB
- Y'CbCr
- Chroma subsampling

Week 3-4
Image Enhancement

Pixel domain
-contrast enhancement
-noise removal

Frequency domain
-DFT
-DCT

Week 5-6
Image Compression
-RGB->Y'CbCr
-Block-wise DCT
-Quantization
-Huffman coding

Forensic Applications

Week 4-5
Digital Watermarking
-pixel domain
-frequency domain

Week 6, 7, 8
Forgery Detection
-compression-based forensics
- Double compression
- Jpeg ghost
-splicing/copy-move forgery detection
-feature matching techniques

Week 8
Source Identification
-sensor pattern noise

Week 9
Invited talks

Week 10
Video forensics

Compression-based forensics

Original JPEG image

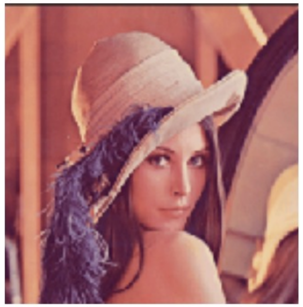
Original JPEG image

Attacker alters the image content

Spliced (JPEG)

Copy-move (JPEG)

JPEG compression



Original

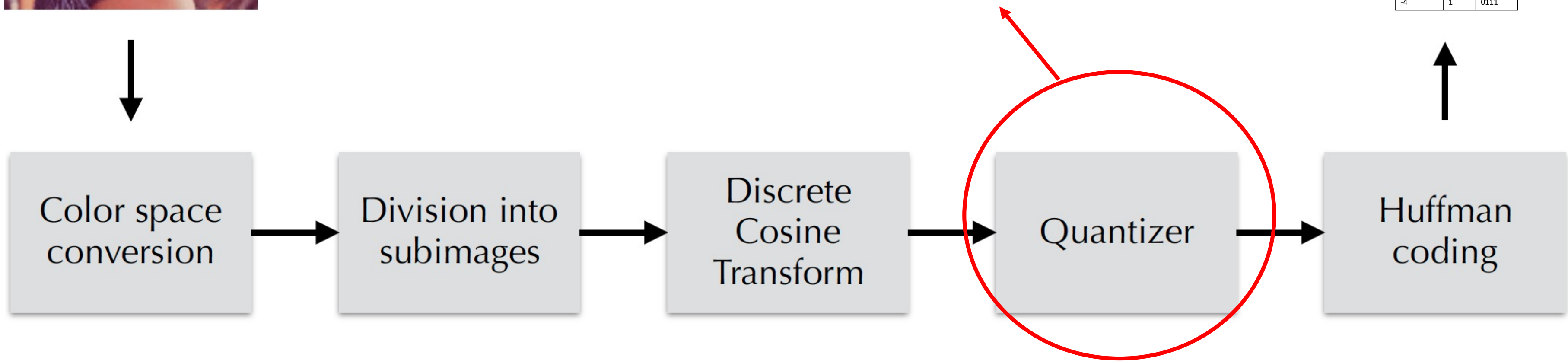
(Bitstream format)

...

value	Freq.	Huffman code
0	56	1
-1	2	000
85	1	0010
5	1	0011
1	1	0100
-2	1	0101
-3	1	0110
-4	1	0111

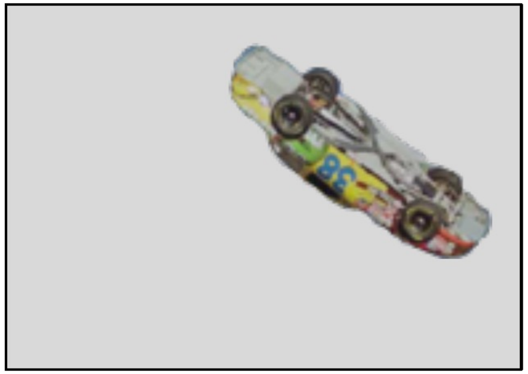
...

The lossy step that significantly changes information content (of course without visual degradation)



Compression-based forensic techniques largely rely on **detecting artifacts introduced due to multiple quantization.**

Splicing forgery



JPEG image (quality = q_1)

JPEG image (quality = q_2)

Quality depends on quantization factors

Region compressed at q_1 then q_3

Rest compressed at q_2 then q_3

Saved as a new JPEG image (quality = q_3)

What happens in copy-move?

JPEG image (quality = q_1)

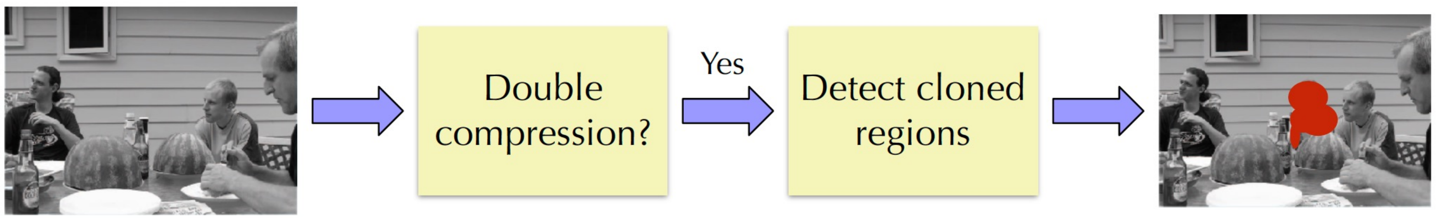
Uncompressed JPEG image, altered using its own parts

Saved as a new JPEG image (quality = q_2)

Compression-based forensics

We will study two popular compression-based forensic techniques. Both the methods try to uncover the multiple quantization artifacts.

- **Double compression**
  - Can only tell us if an image has been compressed multiple times. This can be intentional or accidental. Forgery is a possibility but not confirmed.
  - Used as a screening process. If unusual artifacts are found, further forensic analysis is needed to confirm forgery.



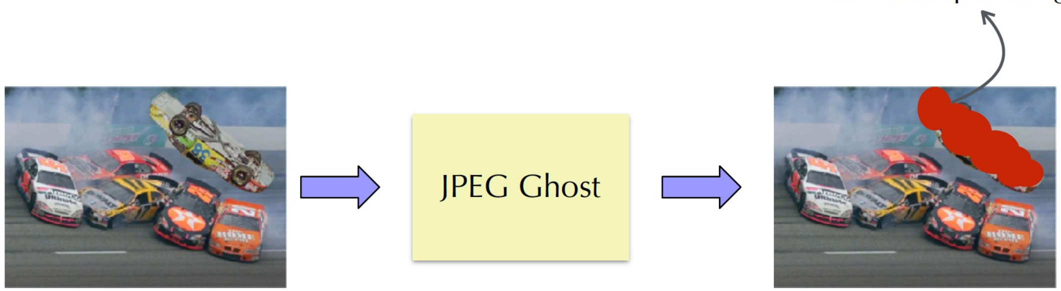
Possibly forged image

Compression-based forensics

We will study two popular compression-based forensic techniques. Both the methods try to uncover the multiple quantization artifacts.

- **JPEG Ghost**
  - can detect splicing forgery, and can even localize the regions that come from a different image.

Detected tampered regions



Possibly spliced image

- Double compression
- JPEG Ghost

Recap: Quantization error

3 X 3 block in range: [0,100]

25	62	15
39	8	77
83	45	58

$F(u,v)$

Fixed quantization:

$Q(u,v) = 5 \; \forall u, \forall v$

Quantized block: $F_Q(u,v)$

5	12	3
8	2	15
17	9	12

$$F_Q(u,v) = \text{round}\left(\frac{F(u,v)}{Q(u,v)}\right)$$

De-quantization

$$\hat{F}(u,v) = F_Q(u,v)Q(u,v)$$

dequantization by the same factor

De-quantized block: $\hat{F}(u,v)$

25	60	15
40	10	75
85	45	60

Quantization error: $\epsilon = |F(u,v) - \hat{F}(u,v)|$

0	2	0
1	2	2
2	0	2

Double Compression/Quantization

Quantization changes DCT coefficients.

Double quantization artifacts will be visible in the **Distribution** of DCT coefficients

Given a variable u , with the entries sampled from a 1D discrete function $f[x]$

Quantization may be considered as an entry-wise operation described by a one-parameter family of functions

$$u \in f[x]$$

$$q_a(u) = \lfloor \frac{u}{a} \rfloor$$

Round or floor operation

Quantization factor (strictly positive integer)

in the example from the previous slide,
it corresponds to $Q(u,v) = 5 \; \forall u, \forall v$

Higher quantization factor indicates low quality

Double Compression/Quantization

Quantization:

$$q_a(u) = \lfloor \frac{u}{a} \rfloor$$

Quantization factor

De-quantization brings the quantized values back to their original range by multiplying **the quantized value** by **the quantization factor**.

$$q_a(u)a = \lfloor \frac{u}{a} \rfloor a$$

A lossy process,
with quantization error

Double Quantization:
(by factor *a* followed by *b*)
a, b are strictly positive integers

$$q_{ab}(u) = \lfloor q_a(u)\frac{a}{b} \rfloor = \lfloor \lfloor \frac{u}{a} \rfloor \frac{a}{b} \rfloor$$

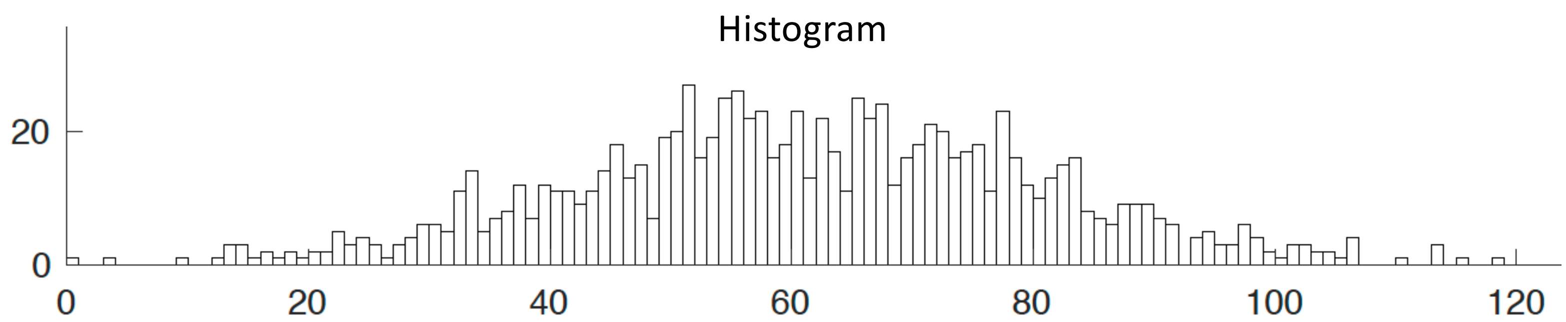
De-quantization by factor a

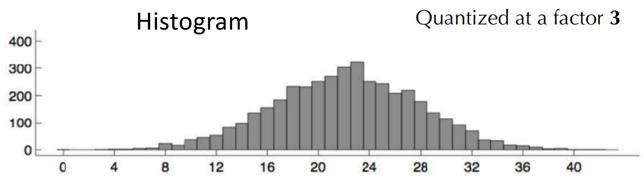
Second quantization by factor b

First quantization by factor a

Example

Consider $f[x]$ to be normally distributed integer samples in the range $[0, 127]$.



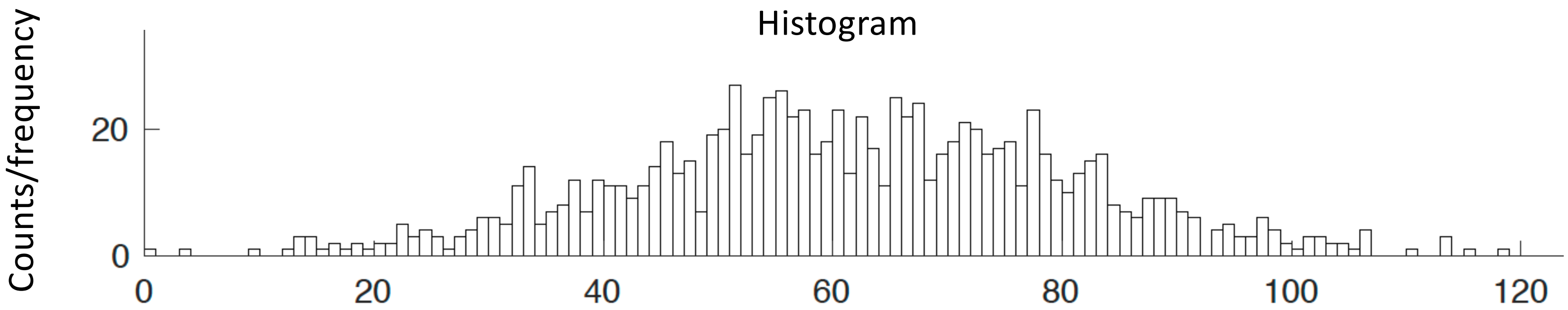


The bins will be updated. For example, original bin 120 will be become bin 40 in new histogram,
How about original bin 121?

Histogram

Counts	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	…
u	0	1	2	3	4	5	6	7	8	9	

This means there are B3 numbers of elements with value 3



Understanding histogram artifacts

Counts	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	…
u	0	1	2	3	4	5	6	7	8	9	

Counts	B0 + B1	B2 + B3	B4 + B5	B6 + B7	B8 + B9	…
floor(u/2)	0	1	2	3	4	

Understanding histogram artifacts

Counts	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	…
u	0	1	2	3	4	5	6	7	8	9	

Counts	B0 + B1	B2 + B3	B4 + B5	B6 + B7	B8 + B9	…
floor(u/2)	0	1	2	3	4	

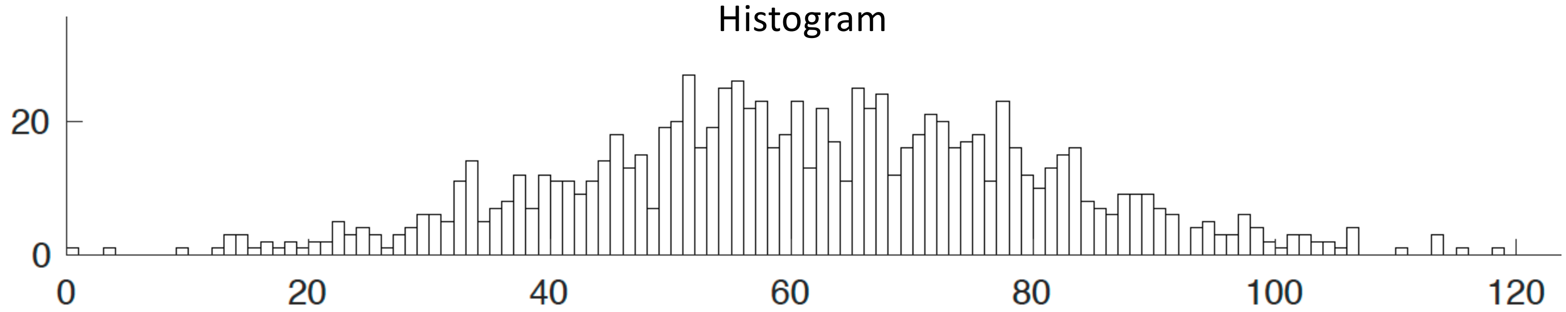
Counts	B0 + B1	B2 + B3	B4 + B5	B6 + B7	B8 + B9	…
*floor(u/2)*2*	0	2	4	6	8	

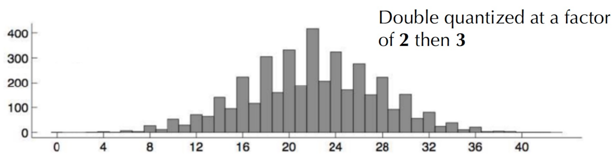
Counts	B0 + B1 + B2 + B3	B4 + B5	B6 + B7 + B8 + B9	B10 + B11	…
*floor(floor(u/2)*2/3)*	0	1	2	3	

Periodic patterns?

Example

Consider $f[x]$ to be normally distributed integer samples in the range $[0, 127]$.





Understanding histogram artifacts

Counts	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	
u	0	1	2	3	4	5	6	7	8	9	…

Counts	B0 + B1 + B2	B3 + B4 + B5	B6 + B7 + B8	B9 + B10 + B11	
floor(u/3)	0	1	2	3	…

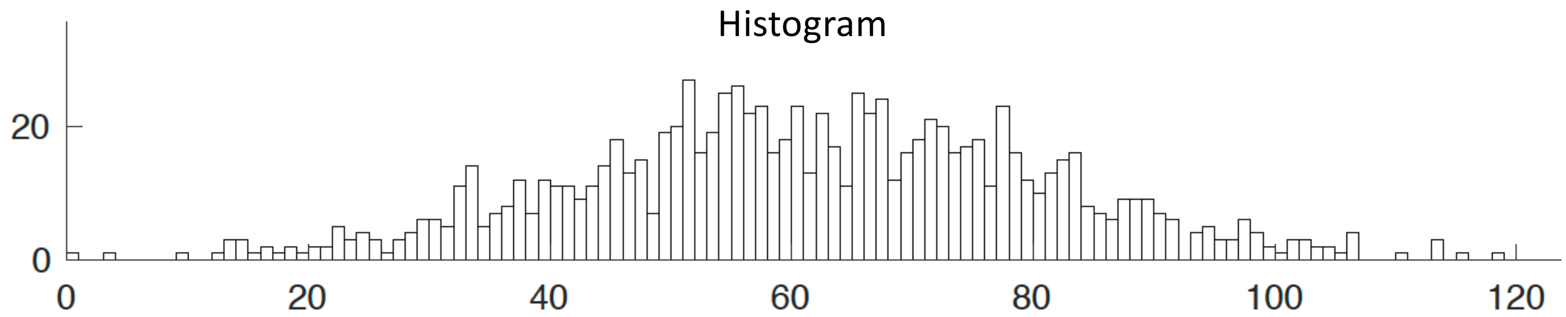
Counts	B0 + B1 + B3	B3 + B4 + B5	B6 + B7 + B8	B9 + B10 + B11	
*floor(u/3)*3*	0	3	6	9	…

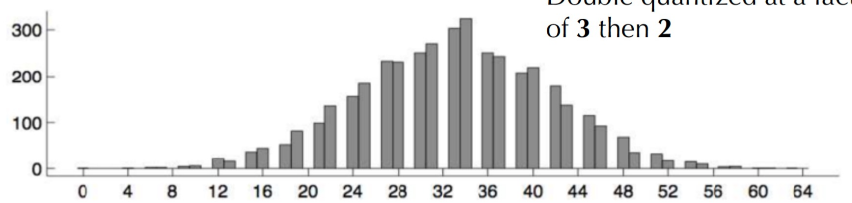
Counts	B0 + B1 + B3	B3 + B4 + B5	**Empty**	B6 + B7 + B8	B9 + B10 + B11	
*floor(floor(u/3)*3/2)*	0	1	2	3	4	…

Periodic patterns?

Example

Consider $f[x]$ to be normally distributed integer samples in the range $[0, 127]$.

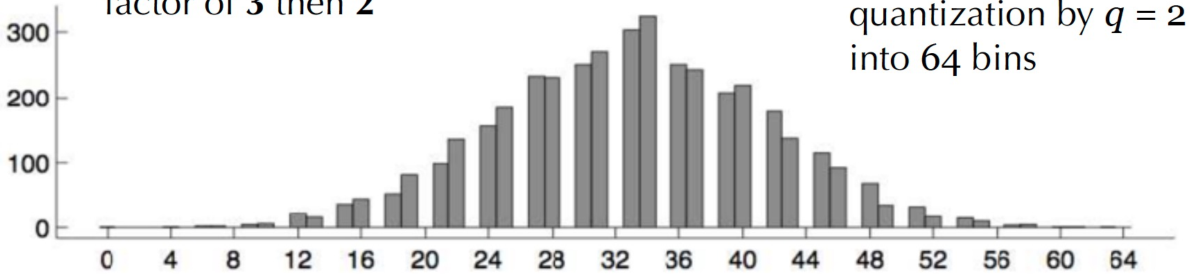




Double quantized at a factor of **3** then **2**

Periodicity

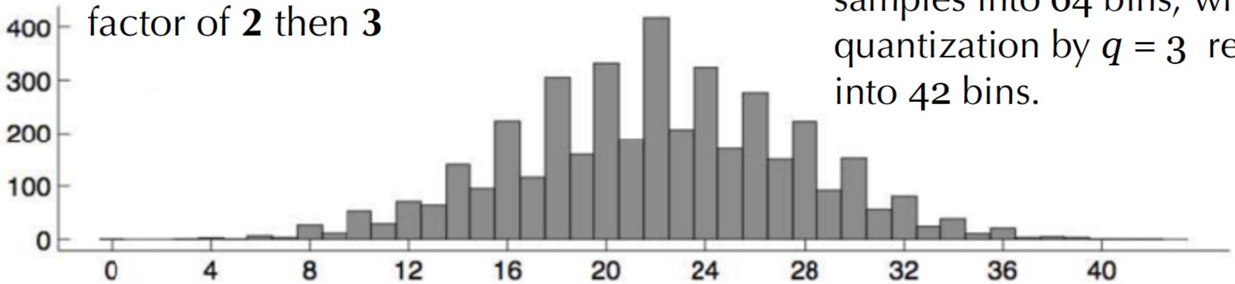
Double quantized at a factor of **3** then **2**



Artifact —> **Periodic empty bins**

quantization by $q = 3$ places the original samples into 42 bins, while the second quantization by $q = 2$ redistributes them into 64 bins

Double quantized at a factor of **2** then **3**



Artifact —> **Periodic peaks**

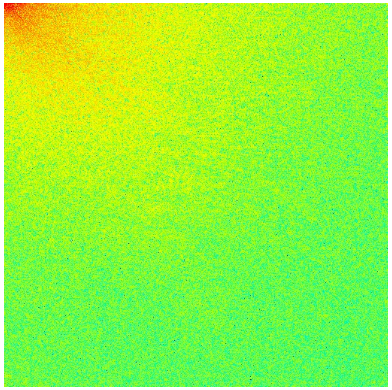
quantization by $q = 2$ places the original samples into 64 bins, while the second quantization by $q = 3$ redistributes them into 42 bins.

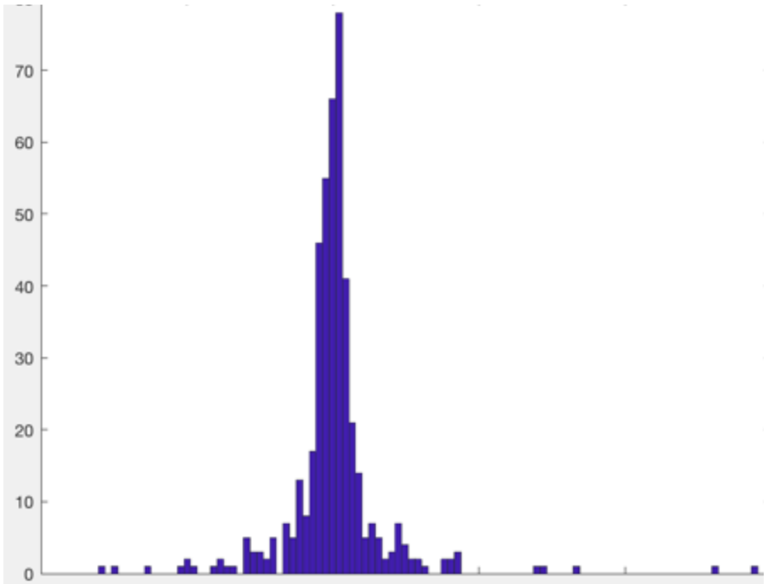
Quantization changes DCT coefficients.

Double quantization artifacts will be visible in the **Distribution** of DCT coefficients

Distribution of DCT coefficients

DCT coefficients
(image-wise)

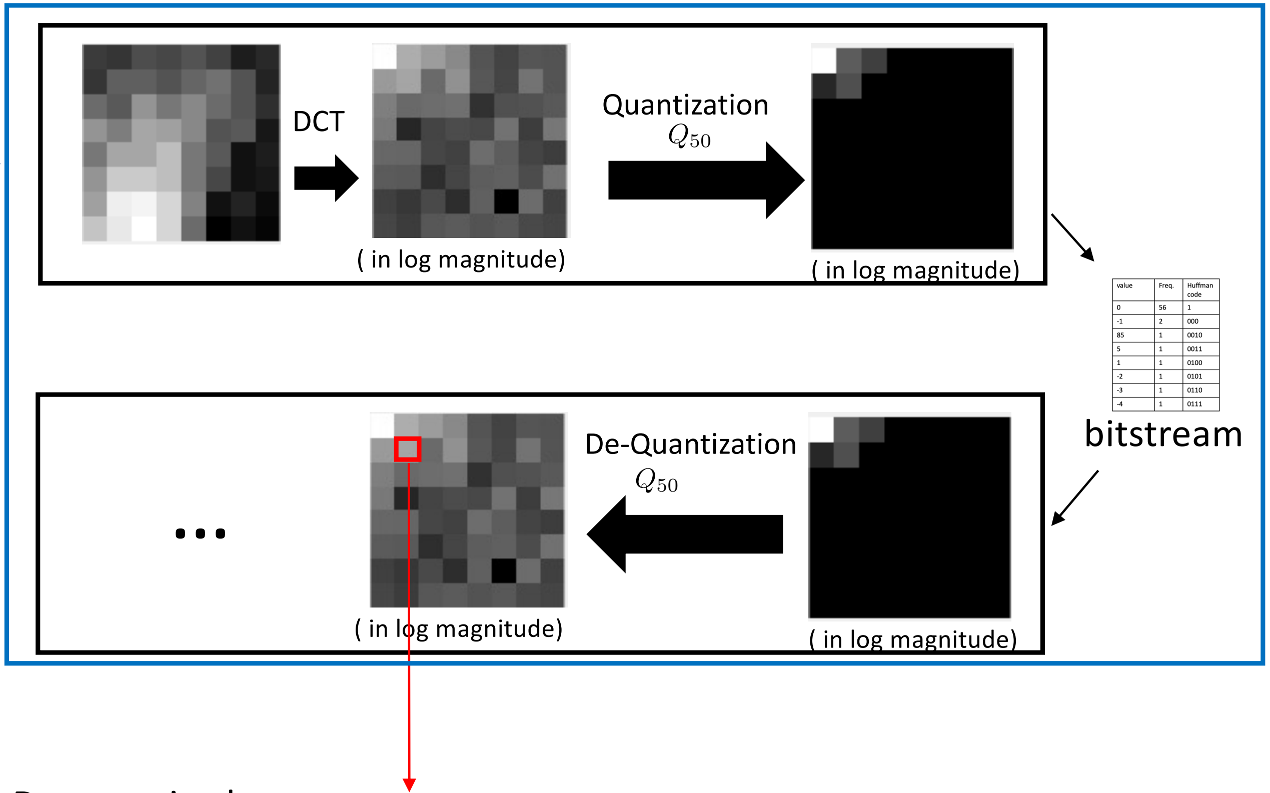




Histogram of mid-frequency DCT coefficients of the Lena image roughly follows a Normal distribution

But it is not Jpeg, and it hasn't been compressed!

(De-quantized) DCT coefficient

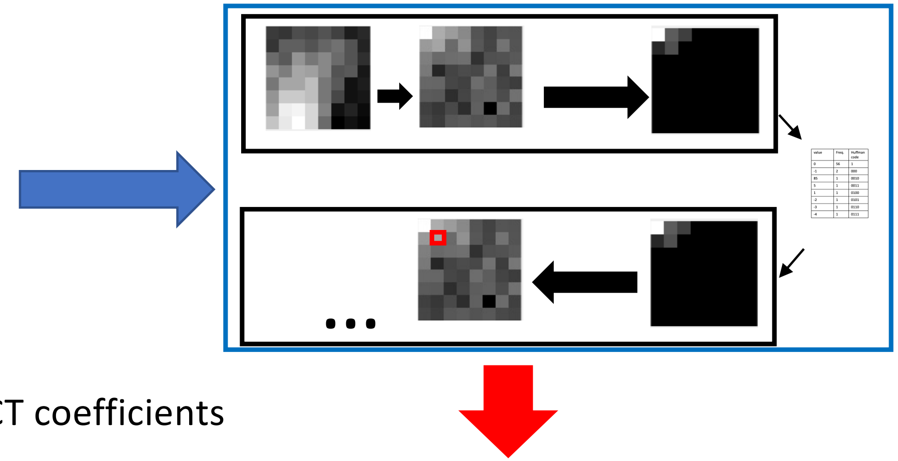


Input: 8*8 blocks

Output: De-quantized
block-wise DCT coefficient

Distribution of DCT coefficients in Jpeg

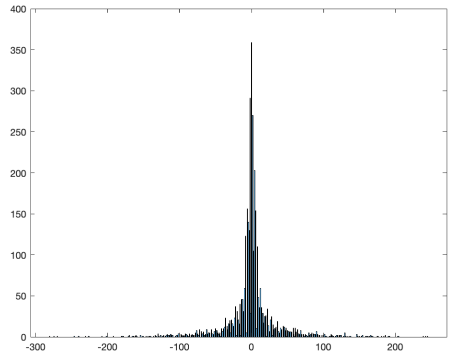
Jpeg compressed with quality 90



Distribution of DCT coefficients for all 8*8 blocks

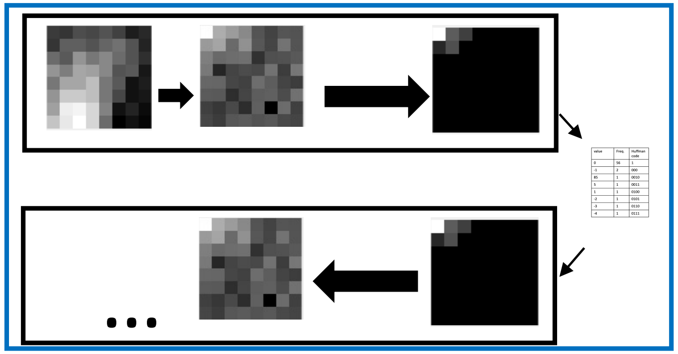
$$\hat{F}(u=1, v=1)$$

(or $\hat{F}(u=2, v=2)$ in Matlab)

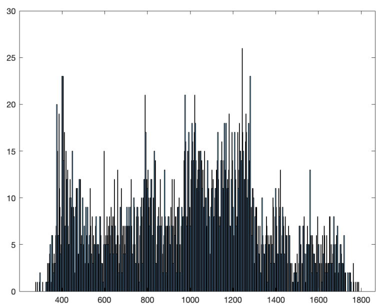


Distribution of DCT coefficients

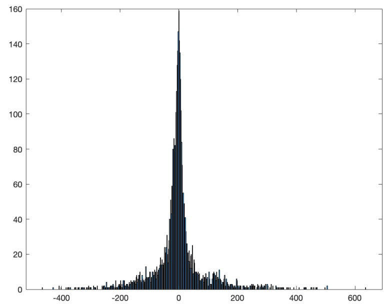
Jpeg compressed with quality 90



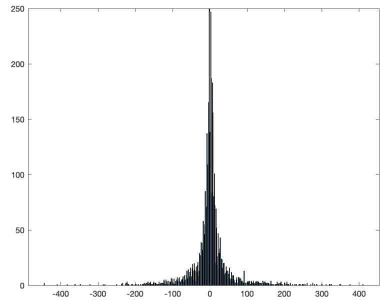
Distribution of DCT coefficients
for all 8*8 blocks



DC

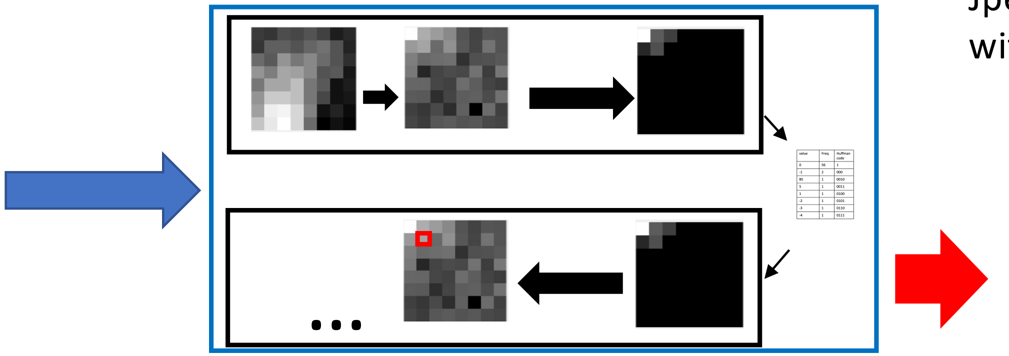


$\hat{F}(u=0, v=1)$



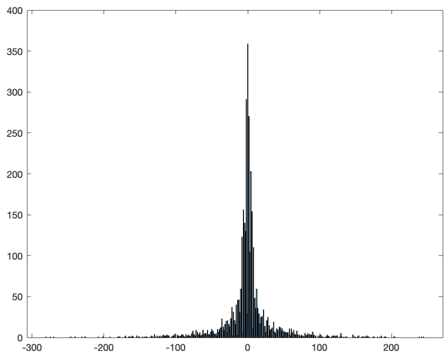
$\hat{F}(u=1, v=0)$

Periodic patterns in double compression

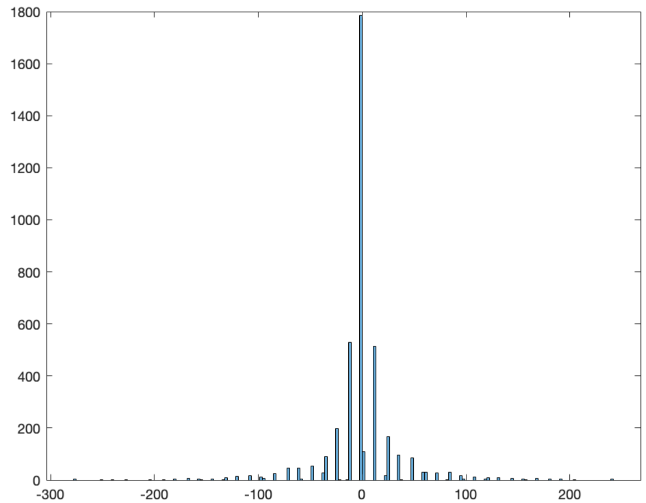


Distribution of DCT coefficients
for all 8*8 blocks

Jpeg compressed
with quality 90



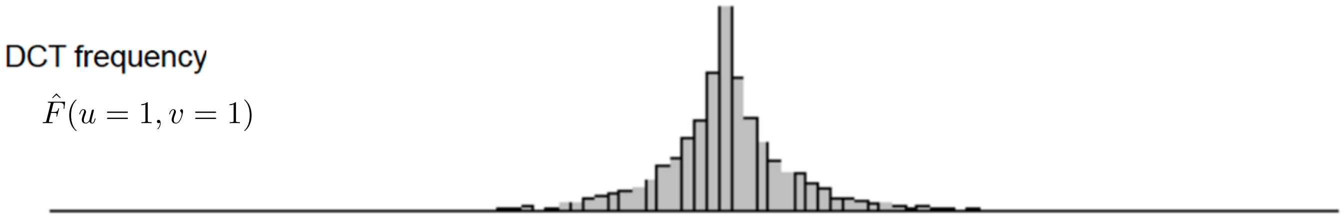
Jpeg compressed
with quality 90, followed by 50



$$\hat{F}(u = 1, v = 1)$$

More Examples

JPEG compressed with
quality 85



JPEG compressed with
quality 75 followed by 85