Cybersecurity Course 2018/2019

Forensic analysis of JPEG image compression

Benedetta Tondi,

University of Siena

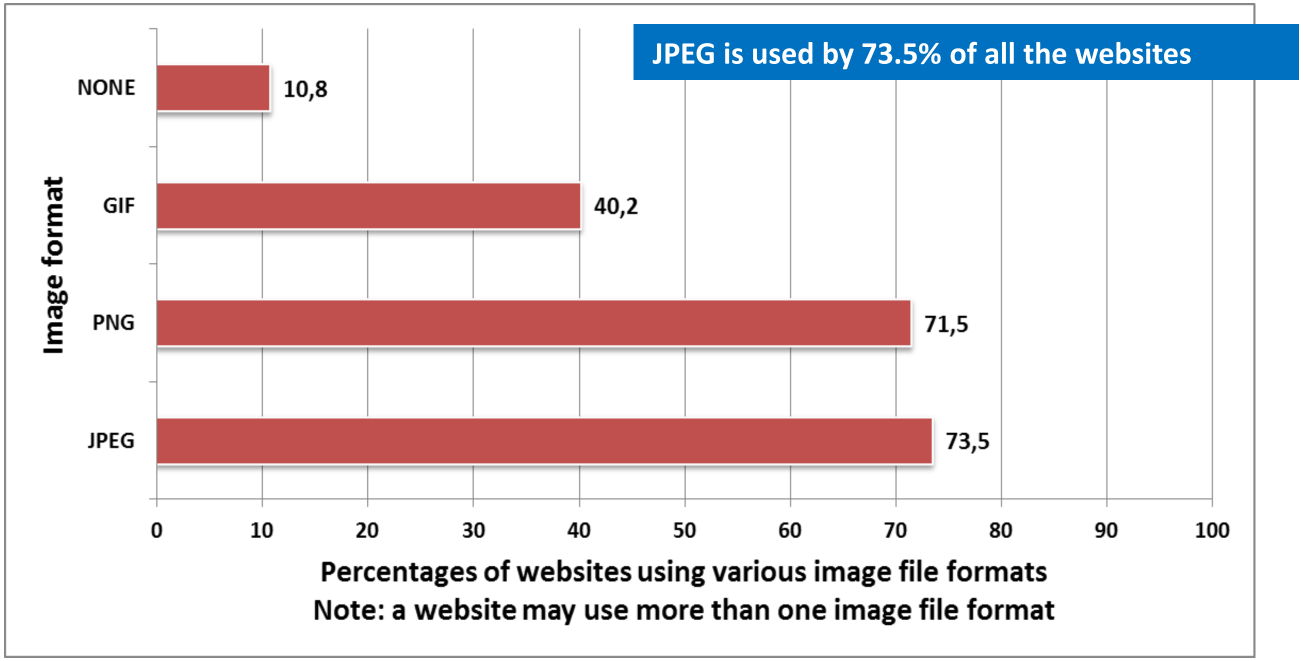
Summary

- Introduction to JPEG
  - What is image compression?
  - The JPEG (Joint Photographic Expert Group) standard
- Forensic analysis of JPEG images
  - Double JPEG image forensics

Introduction

What is JPEG?

- JPEG (Joint Photographic Expert Group) is an international standard for **lossy image compression** released in 1992
- JPEG is still today one of the most popular image formats on the Web

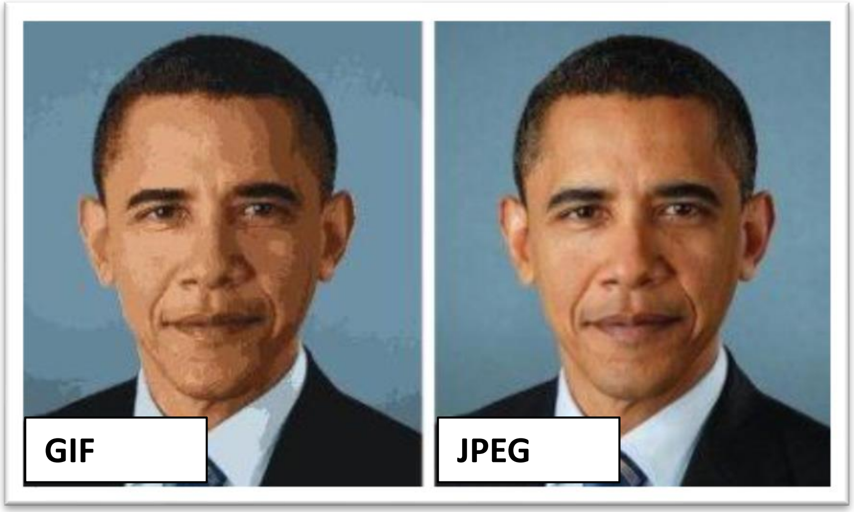


Source: https://w3techs.com/technologies/overview/image_format/all (updated April 2016)

- Photos in social networks are in (lossy) compressed formats. Most of them are in JPEG format

What is JPEG?

- JPEG is used in many applications. It is particularly suitable for the compression of **photos and paintings of realistic scenes** with smooth variations of tone and color



- With respect to the also widely diffused GIF format, JPEG ensures **better visual quality** compressed images **for the same file size**

Importance of compression (in real life)

File type	Size
Tiff, uncompressed	901K
Tiff, LZW lossless compression (yes, its actually bigger)	928K
JPG, High quality	319K
JPG, medium quality	188K
JPG, typical web quality	105K
JPG, low quality / high compression	50K
JPG, absurdly high compression	18K
PNG, lossless compression	741K
GIF, lossless compression, but only 256 colors	286K

Impact of compression (in real life)

Higher compression at the pice of a lower visual quality

Why/How can images be compressed?

- Image compression can be achieved because image data are **often hightly redundant and/or irrelevant.**
- Image coding is achieved by *reducing the redundancy* contained in data. More specifically, two kinds of redundancy exist:
  - *statistical redundancy*, which is exploited for lossless compression
  - *Irrelevance (psychovisual redundancy)*, whose removal leads to lossy compression

Statistical redundancy

- Spatial redundancy
  - correlation between neighboring pixels
- Spectral redundancy
  - correlation among color components
- Temporal redundancy (for video compression)
  - correlation between consecutive frames

Spatial redundancy: an example

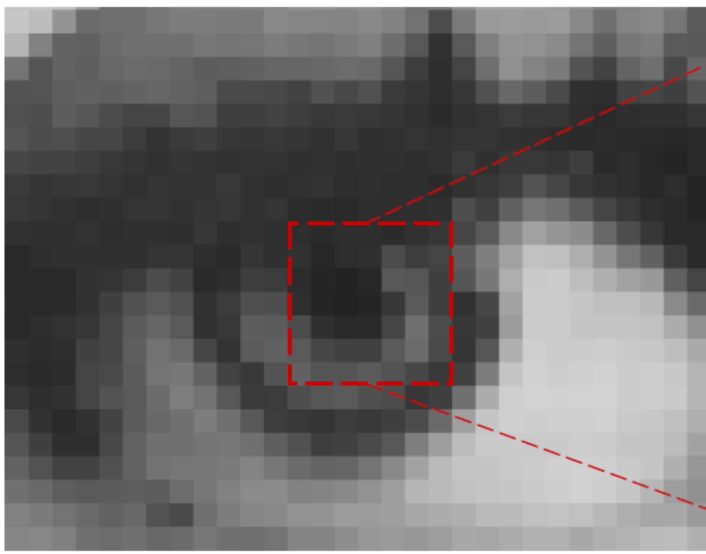
- The difference between two adjacient pixels has a very skewed distribution centered around 0

Psychovisual redundancy

- Spatial irrelevance
  – refers to the ability of the Human Visual System (HVS) to perceive small image details
- Spectral irrelevance
  – refers to the way the HVS perceives colors
- Temporal irrelevance (for video compression)
  – accounts for the ability of the HVS to perceive rapid changes between subsequent video frames

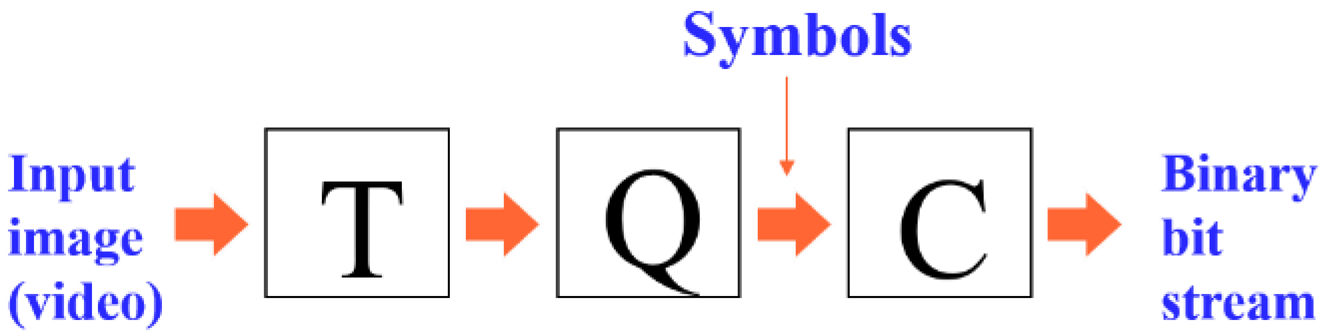
Spatial redundancy and…..irrelevancy

- What is the value of the missing pixel? **(39)**
- How critical is the exact reproduction?



46	44	46	47	44	50	56
50	40	46	48	76	62	69
45	39	36	40	88	87	65
48	38	36	?	56	90	65
76	48	41	43	69	112	77
90	72	66	66	90	108	74
98	86	86	91	83	72	57

General compression scheme

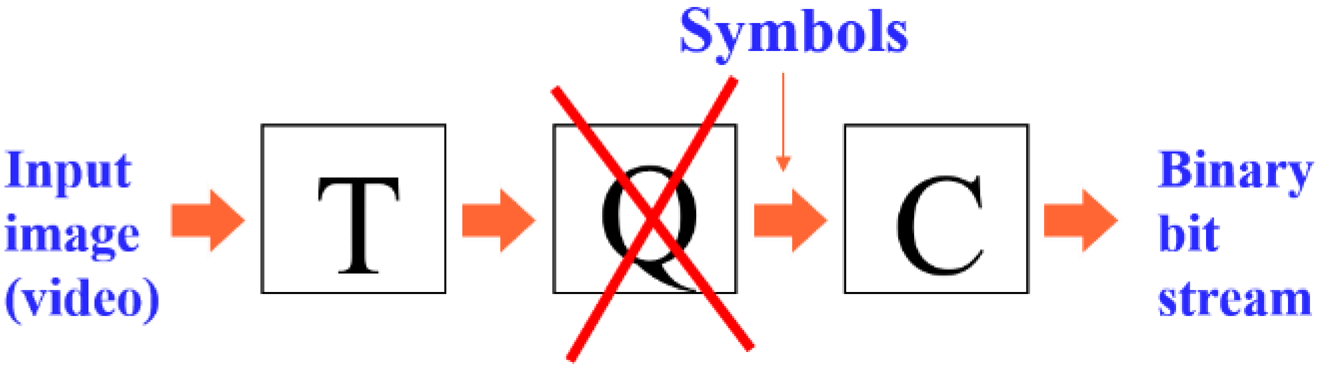


- **T = Transformer**, it applies a one-to-one transformation to input data, the output should be more amenable to compression (e.g. skew probability distribution, reduced correlation among data). No loss occurs here. Examples: predictive mapping, DCT transform.

- **Q = Quantizer**, it achieves lossy compression by performing a many-to-one mapping of data into symbols (scalar or vector quantization)

- **C = Coder**, by assigning a codeword to each symbol produced by the quantizer, lossless compression is achieved (Fixed-lenght or variable-lenght codes may be used)

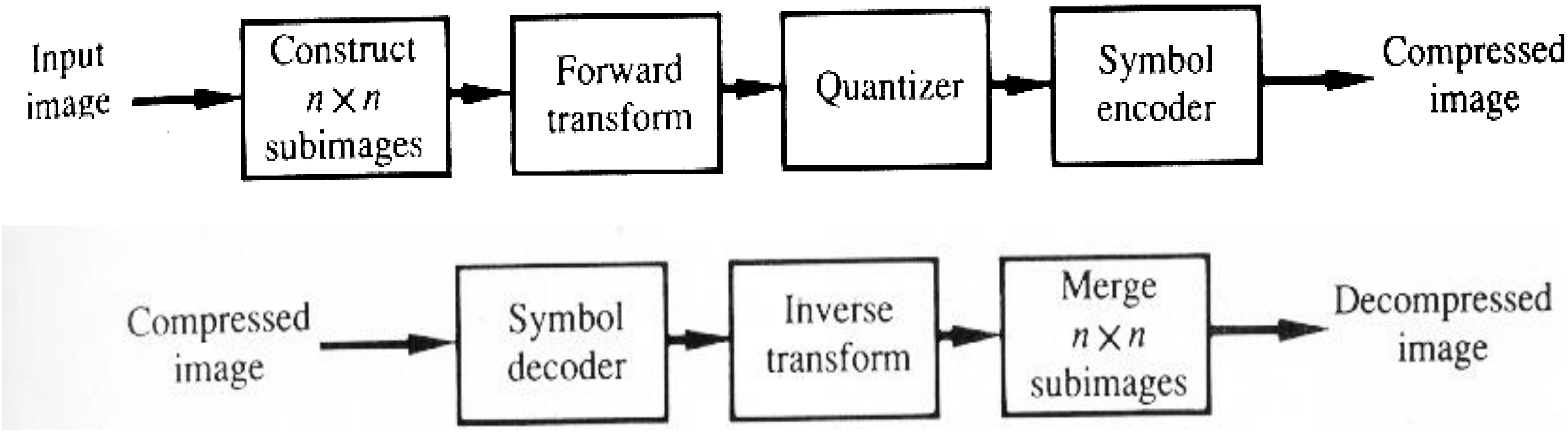
Lossless compression



- By removing the quantizer, a general lossless compression scheme is obtained
- The transformer **T** only aims at removing the spatial, spectral and temporal redundancy (memory), or at putting it in a different form, so that it is easier for the symbol coder to compress data
- *Compression ratios achievable through lossless coding are not sufficient to meet the needs of most practical applications*

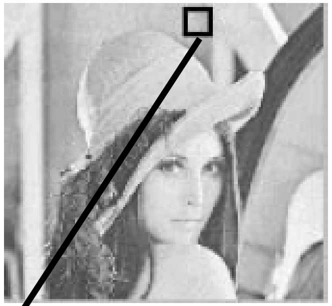
Block-based transform (T)

- Transform coding is performed by taking an image and breaking it down into sub-image (block) of size nxn. The transform is then applied to each sub-image (block) and the resulting transform coefficients are quantized and entropy coded.



An example

8x8 image block of the image of Lena



DCT transform of the 8x8 block

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i,j)$

T

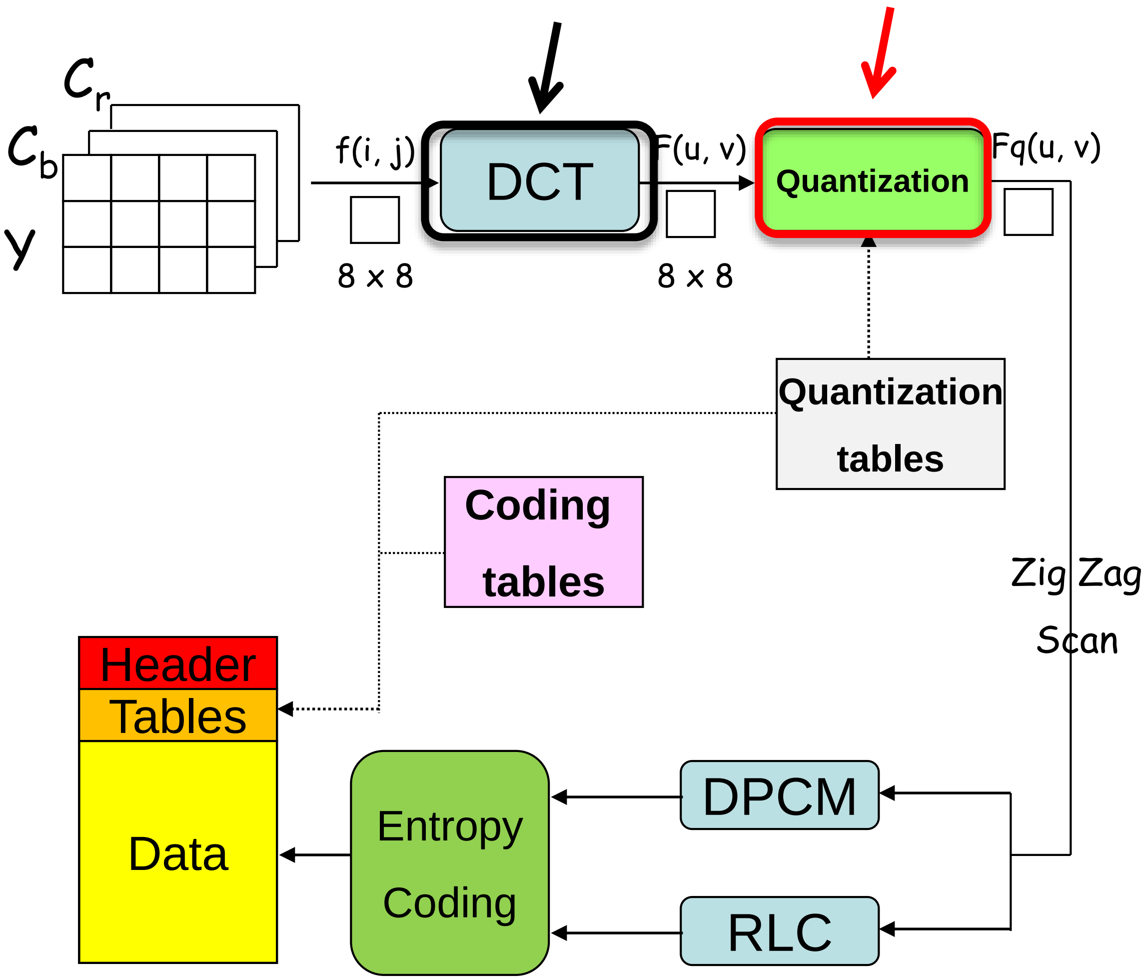
515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u,v)$

(Smooth block !)

Most of the energy contained in few coefficients!

JPEG baseline encoding



Main steps:

1. **Discrete Cosine Transform of each 8x8 pixel block**
2. **Scalar quantization**
3. Zig-zag scan to exploit redundancy
4. Data Preparation for Entropy coding (DPCM, RLC)
5. Entropy coding

Reverse order for decoding

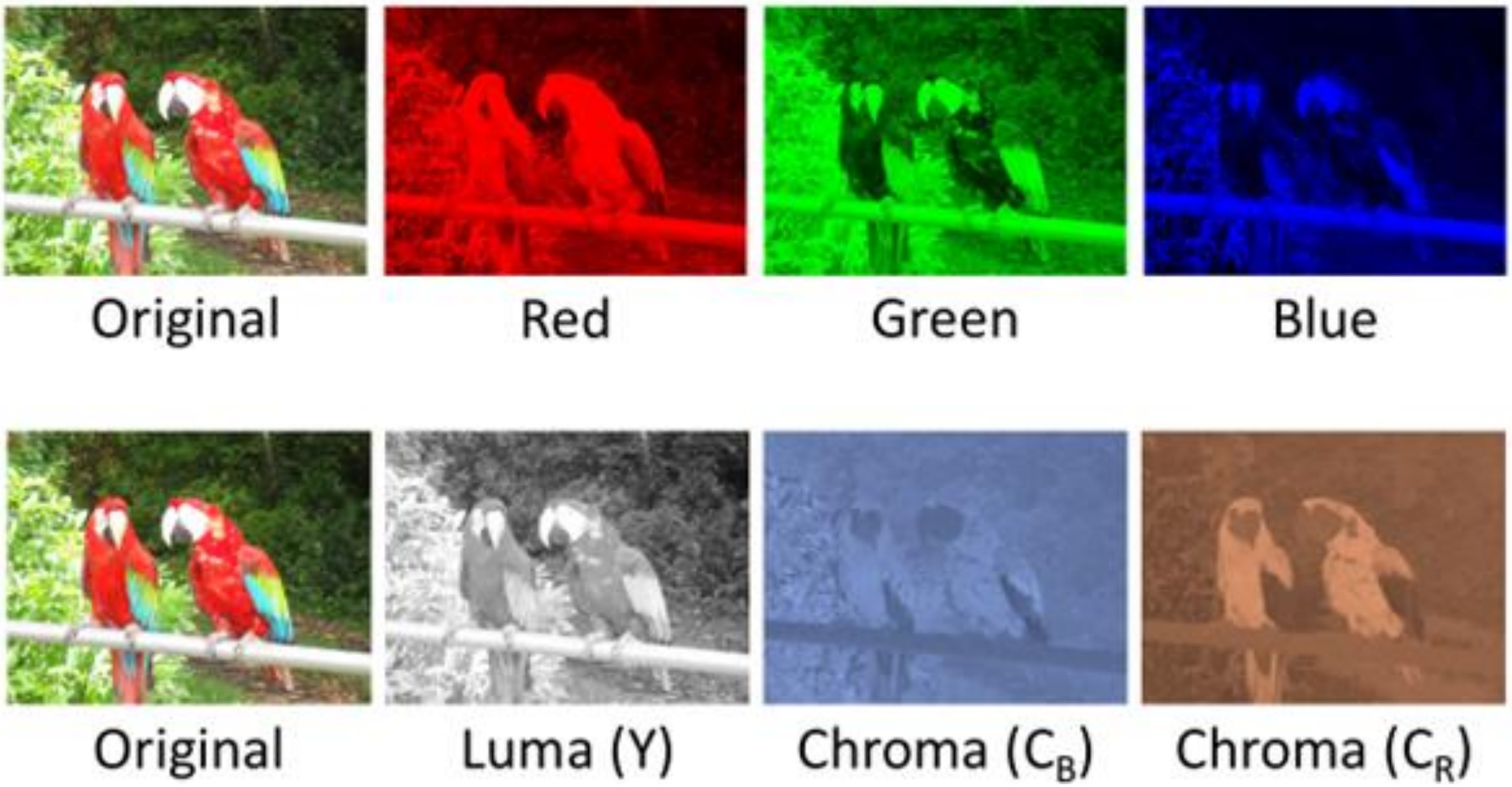
Color space transform: RGB to YCbCr

- **RGB** color space is not the only method to represent an image
- There are several other color spaces, each one with its properties
- A popular color space in image compression is the **YCbCr**, which:
  - separates *luminance* (Y) from *color information* (Cb,Cr)
  - processes Y and (Cb,Cr) separately (not possible in RGB !)
- RGB to YCbCr (and YCbCr to RGB) linear conversions:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \qquad \begin{matrix} Y \in [0, 255] \\ C_b \in [0, 255] \\ C_r \in [0, 255] \end{matrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.403 \\ 1.000 & -0.344 & -0.714 \\ 1.000 & 1.773 & 0.000 \end{bmatrix} \cdot \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix} \qquad \begin{matrix} R \in [0, 255] \\ G \in [0, 255] \\ B \in [0, 255] \end{matrix}$$

Color space transform – example

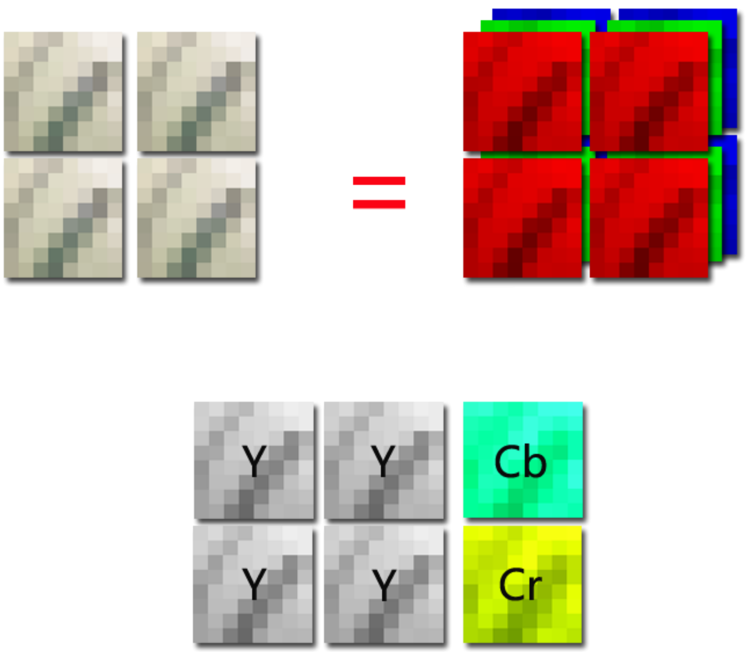


Color space transform – subsampling

- Y is taken every pixel, and Cb,Cr are taken for a block of 2x2 pixels

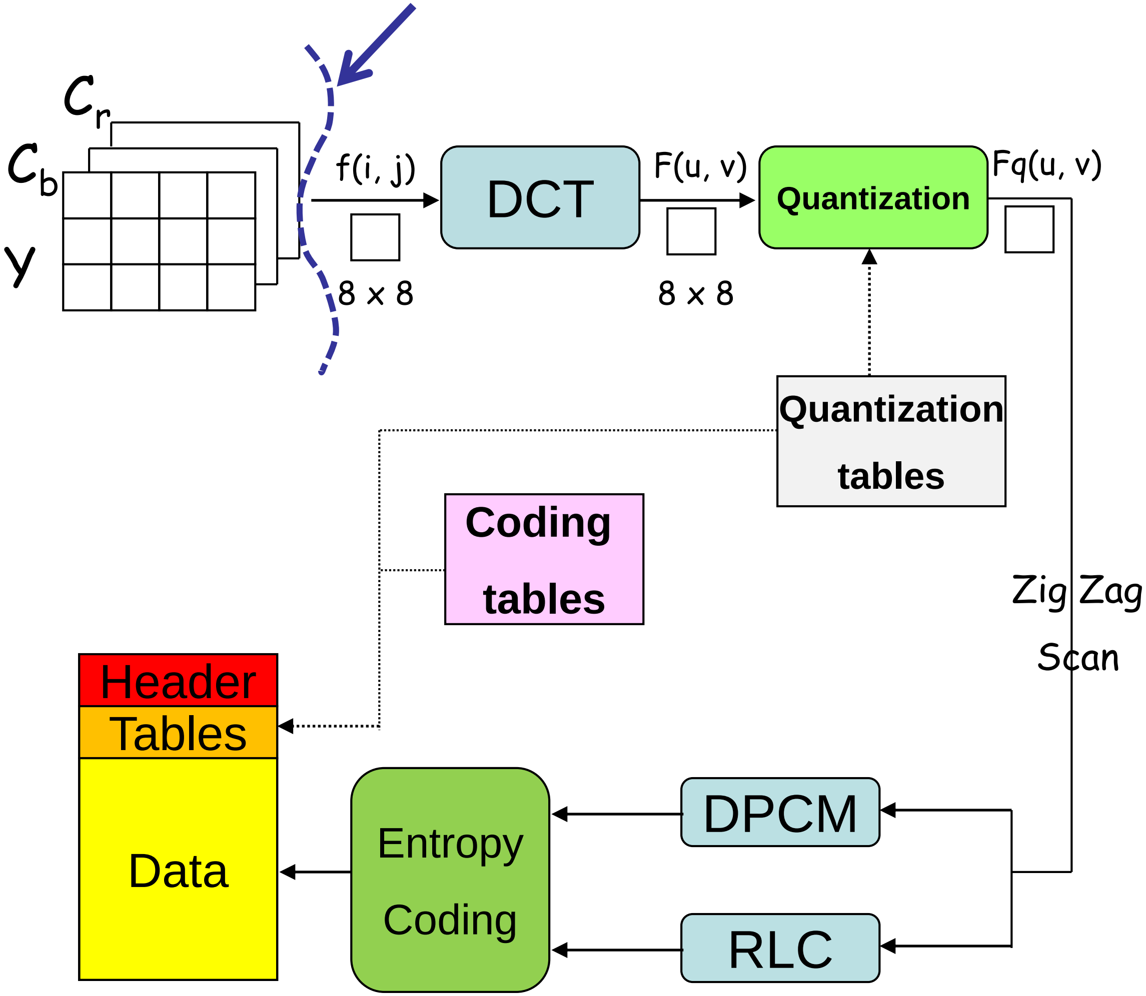
Data size is reduced to a half without significant losses in visual quality

- Example: block 64x64



Without subsampling, one must take 64^2 pixel values for each color channel: **$3 * 64^2 = 12288$ values (1 bytes per value)**
JPEG takes 64^2 values for Y and $2x32^2$ values for chroma **$64^2 + 2x32^2 = 6144$ values (1 bytes per value)**

JPEG baseline encoding



Main steps:

1. Discrete Cosine Transform of each 8x8 pixel block
2. Scalar quantization
3. Zig-zag scan to exploit redundancy
4. Data Preparation for Entropy coding (DPCM, RLC)
5. Entropy coding

Reverse order for decoding

Discrete Cosine Transform (DCT)

- **Transformed data are more suitable to compression** (e.g. skew probability distribution, reduced correlation).
- 2D-DCT

Forward DCT

$$F(u,v)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right]$$

for $u=0,\ldots,7$ and $v=0,\ldots,7$

$$\text{where } C(k)=\begin{cases}1/\sqrt{2} & \text{for } k=0\\ 1 & \text{otherwise}\end{cases}$$

Inverse DCT

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right]$$

for $x=0,\ldots,7$ and $y=0,\ldots,7$

2D-DCT: computation

Pixel block

214	224	199	190	222	231	239	236
215	198	179	218	219	223	229	236
202	166	205	215	218	208	144	183
173	177	211	212	200	152	136	196
155	197	205	200	143	133	178	221
158	197	198	153	118	159	202	211
169	192	167	110	139	190	196	199
169	180	131	105	172	193	190	191

$- 128 =$

86	96	71	62	94	103	111	108
87	70	51	90	91	95	101	108
74	38	77	87	90	80	16	55
45	49	83	84	72	24	8	68
27	69	77	72	15	5	50	93
30	69	70	25	−10	31	74	83
41	64	39	−18	11	62	68	71
41	52	3	−23	44	65	62	63

Shift operations

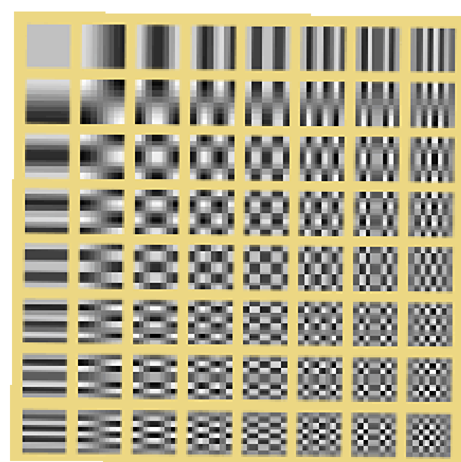
From [0, 255]

To [-128, 127]

$$S_{vu} = \frac{1}{4} C_u \ C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

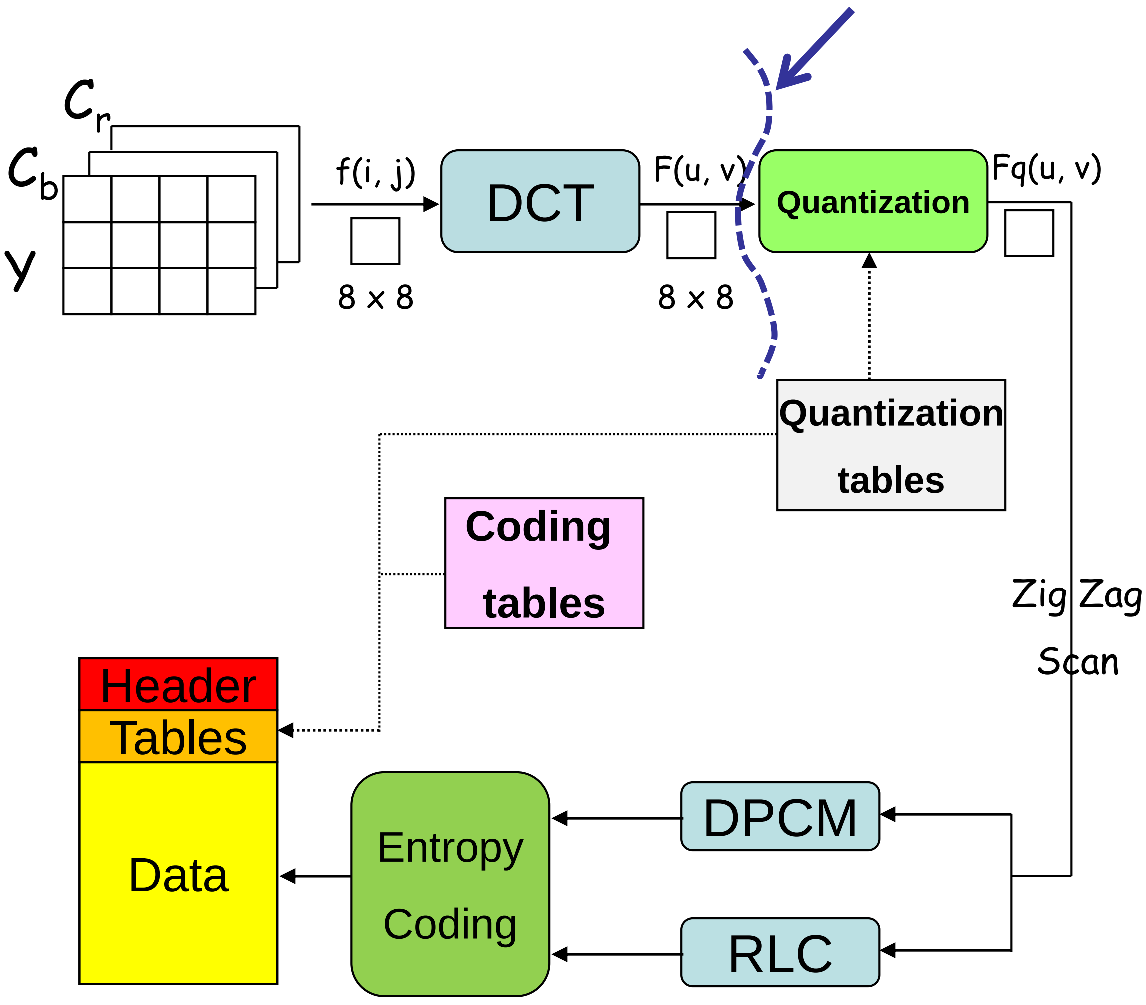
DCT Result

479	−35	52	−30	−7	−27	10	−3
141	11	−62	11	56	14	5	−5
41	−54	48	98	−35	−19	−14	8
22	−16	56	−48	−48	17	−24	−3
−4	5	−6	−16	14	−15	−15	9
−10	20	−11	9	−7	−15	22	12
−9	12	−12	5	−4	0	14	3
−2	6	−2	3	−3	0	8	−2



Meaning of each position in DCT result-matrix

JPEG baseline encoding



Main steps:

1. Discrete Cosine Transform of each 8x8 pixel block
2. Scalar quantization
3. Zig-zag scan to exploit redundancy
4. Data Preparation for Entropy coding (DPCM, RLC)
5. Entropy coding

Reverse order for decoding

Quantization

- **Goal: to reduce number of bits per sample**

- For each 8x8 DCT block, *F(u.v)* is divided by a 8x8 **quantization matrix *Q***

$$\boxed{F_q(u,v) = \mathrm{round}\left(\frac{F(u,v)}{Q(u,v)}\right)}$$

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

(*Reconstructed value*)

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

(*Reconstruction error*)

Q(u,v), *quantization step* at frequency (u,v)

- Example (one number): F = 45
  - Q= 4: F_q = round(11.25) = 11 (De-quantize: 11x4 = 44, against 45. **Err = 1**)
  - Q= 8: F_q = round(5.625) = 6 (De-quantize: 6x8 = 48, against 45. **Err = 3**)

- **Quantization error is the main reason why JPEG compression is LOSSY**

Quantization

- Each *F[u,v]* in a 8x8 block is divided by constant value *Q(u,v).*
- **Higher values in the quantization matrix Q allows to achieve *better compression* at the cost of *visual quality***
- **How to choose Q?**
- Eye is *more sensitive to low frequencies* (upper left corner of the 8x8 matrix), *less sensitive to high frequencies* (lower right corner)….

Quantization

- Each *F[u,v]* in a 8x8 block is divided by constant value *Q(u,v).*
- **Higher values in the quantization matrix Q allows to achieve *better compression* at the cost of *visual quality***
- **How to choose Q?**
- Eye is *more sensitive to low frequencies* (upper left corner of the 8x8 matrix), *less sensitive to high frequencies* (lower right corner)….
- **Idea: quantize more (large quantization step) the high frequencies, less the low frequencies**
- The values of the Q matrix are controlled with a parameter called **Quality Factor (QF).**
  - **QF ranges from 100 (best quality) to 1 (extremely low)**

Quantization table: luminance

- Example: Quantization table Q for **QF = 50**

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization table: chrominance

- Example: Quantization table Q for **QF = 50**
- Colors can be quantized more coarsely due to reduced sensitivity of the Human Visual System (HVS)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Quantization: luminance and chrominance

- An example of quantization table Q for **QF = 70**

Quantization Table: Luminance

10	7	6	10	14	24	31	37
7	7	8	11	16	35	36	33
8	8	10	14	24	34	41	34
8	10	13	17	31	52	48	37
11	13	22	34	41	65	62	46
14	21	33	38	49	62	68	55
29	38	47	52	62	73	72	61
43	55	57	59	67	60	62	59

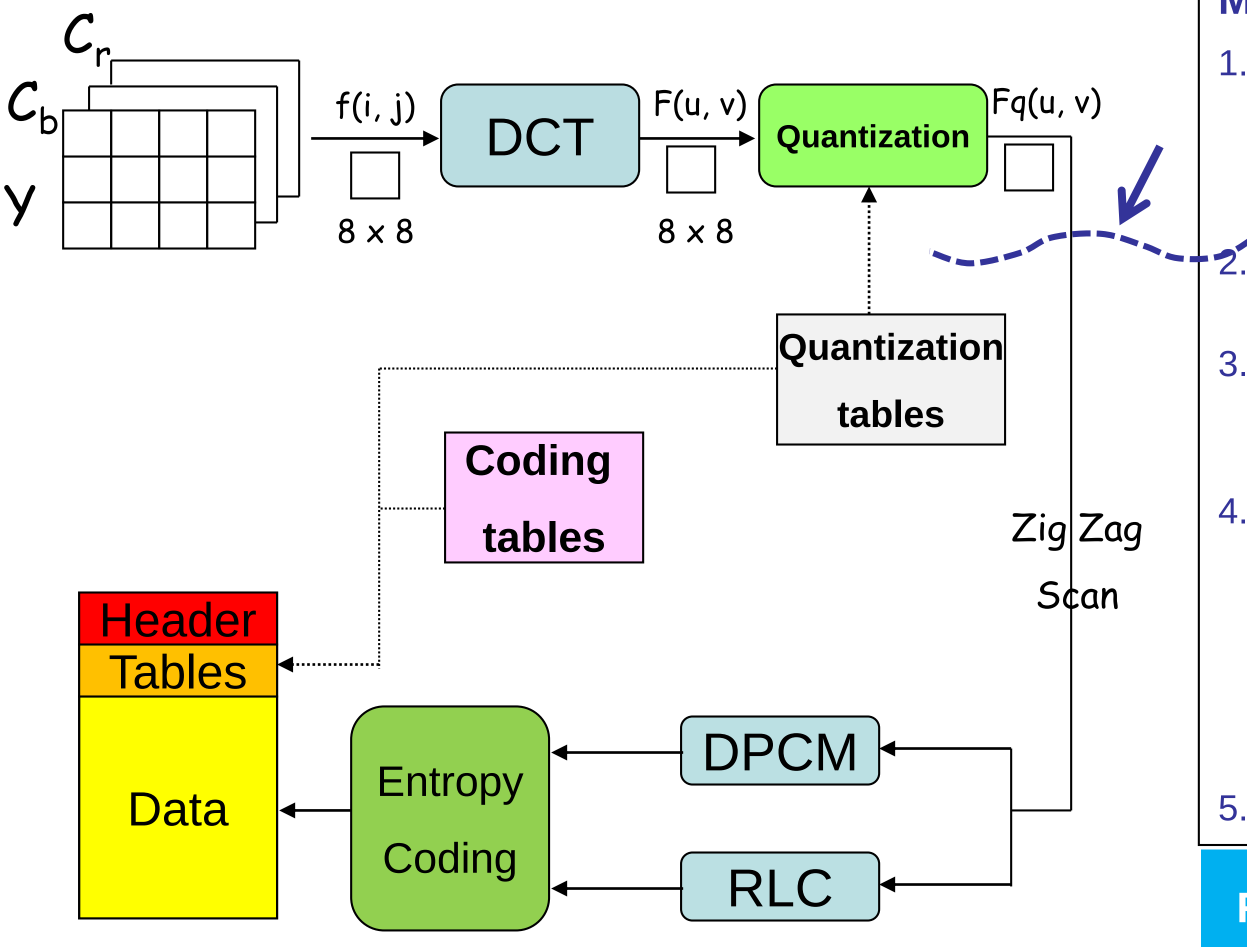
Quantization Table: Chrominance

10	11	14	28	59	59	59	59
11	13	16	40	59	59	59	59
14	16	34	59	59	59	59	59
28	40	59	59	59	59	59	59
59	59	59	59	59	59	59	59
59	59	59	59	59	59	59	59
59	59	59	59	59	59	59	59
59	59	59	59	59	59	59	59

- The quantization is less strong at larger QF


NO JPEG (20MB)
JPEG 100 (9MB)
JPEG 60 (1.3MB)
JPEG 20 (0.6MB)
JPEG 5 (0.4MB)

JPEG baseline encoding



Main steps:

1. Discrete Cosine Transform of each 8x8 pixel block
2. Scalar quantization
3. Zig-zag scan to exploit redundancy
4. Differential Pulse Code Modulation (DPCM) on the DC component and Run Length Encoding of the AC components
5. Entropy coding (Huffman)

Reverse order for decoding

Preparation for Entropy Coding

- We have seen two main steps in JPEG coding: DCT transform (T) and quantization (Q)
- The remaining steps all lead up to entropy coding (C) of the *quantized block-DCT coefficients*
  - These additional data compression steps are *lossless*
  - Most of the lossiness is in the quantization step

Remarks on JPEG compression

JPEG is effective because of the following main points:

- *Image data usually changes slowly across an image, especially within an 8x8 block*
  - Therefore images contain *much redundancy*
- *Experiments indicate that humans are not very sensitive to the high frequency data images*
  - Therefore we can remove much of this data exploiting transform coding
- *Humans are much more sensitive to brightness (luminance) information than to color (chrominance)*
  - JPEG performs subsampling of chrominance information (color channels)

Forensic Analysis

of JPEG images

JPEG compression footprints

- Like any other image processing, **JPEG leaves traces into the image**, especially at low Quality Factors
  - Such traces can be exploited to **gather useful information on the image**

- Some JPEG artifacts are immediately identified
  - **Blocking** due to block discontinuities
  - **Ringing** on edges due to the DCT
  - **Graininess** due to coarse quantization
  - **Blurring** due to high frequency removal

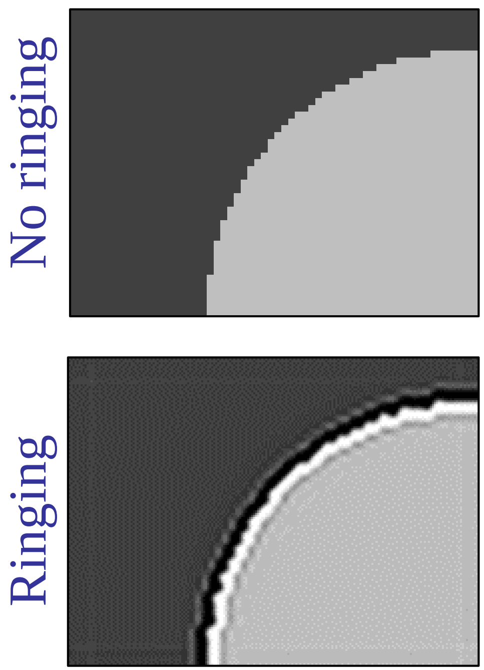
- **Other (statistical) alterations are more subtle to identify!**

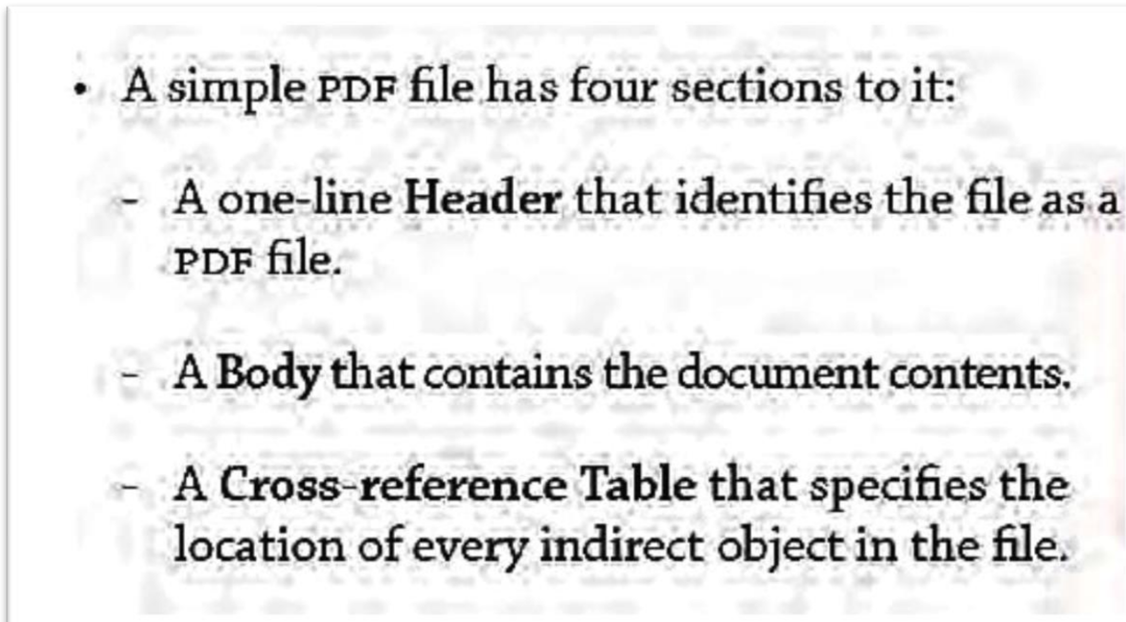
Blocking artifacts

- Processing each 8x8 block independently introduces **discontinuities along the block boundaries**, thus making image tiling visible

Ringing artifacts

- Spurious signals **near sharp transitions**
  - Visually, they appear as bands or “ghosts”
  - Particularly evident along edges an in text images





- A simple PDF file has four sections to it:
  - A one-line **Header** that identifies the file as a PDF file.
  - A **Body** that contains the document contents.
  - A **Cross-reference Table** that specifies the location of every indirect object in the file.

Graininess artifacts

- Particularly evident as “dots” along the edges

Blurring artifacts

- Removing high frequency DCT coefficients increases the smoothness of the image, retaining shapes but making textures less distinguishable
  - **Human eye is particularly good at spotting smoothness**

Double JPEG compression forensics

Double JPEG compression forensics

- Double JPEG compression is when an image is JPEG compressed first with QF_1 and then JPEG compressed again with QF_2

- In **MM-Forensics**, several approaches have been proposed to reveal **the footprints left by double compression**

Why understanding whether an image has been JPEG compressed (quantized) twice is important?

Suppose you took this nice picture with your camera. Image that this picture did not undergo any compression (a TIF image, for example)

Download an image from the Internet. It is very likely that this one is a JPEG file, that is, the image is JPEG compressed with a certain QF

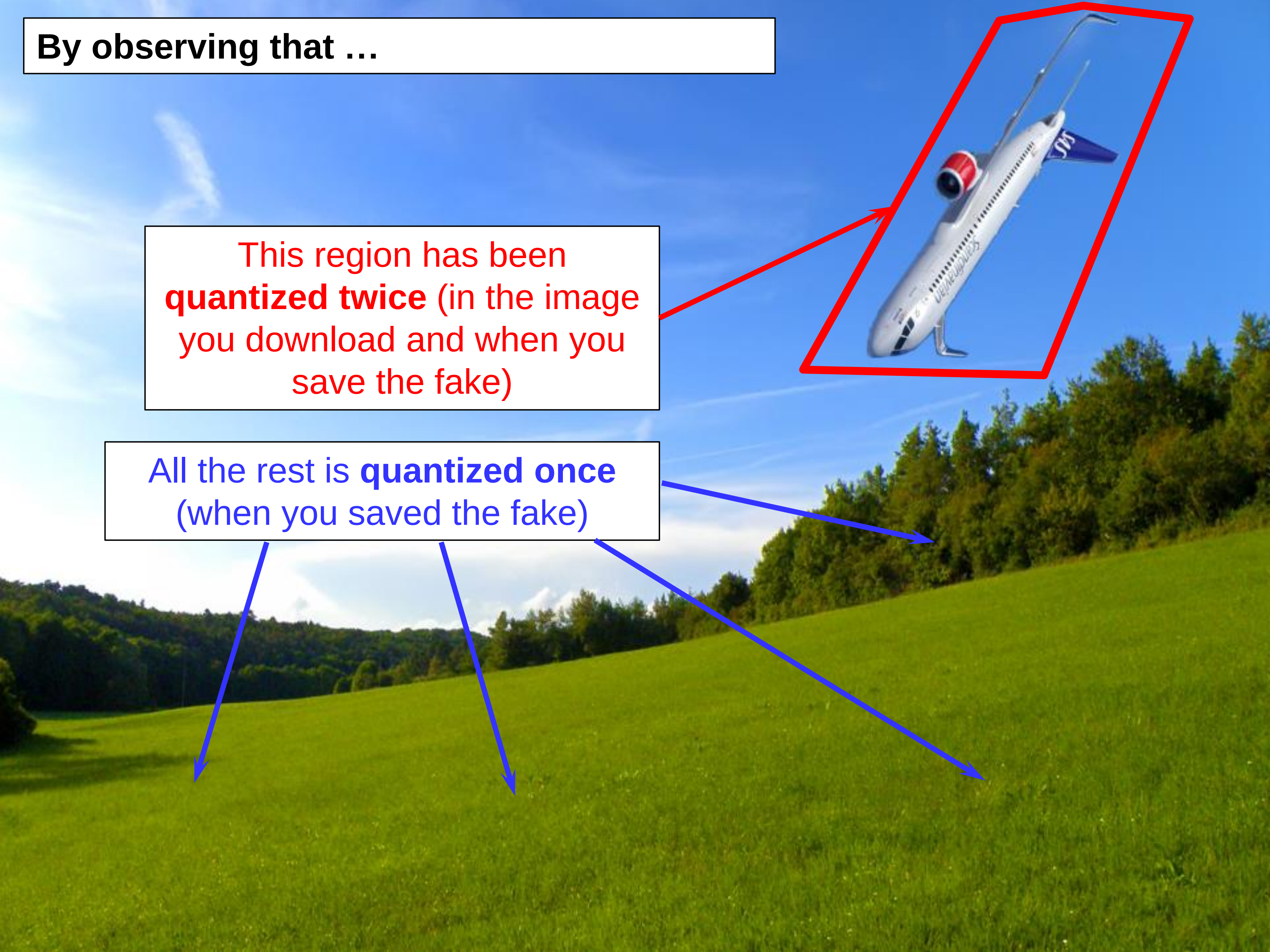
Start your favorite image editing software ….

Create a fake, realistic and deceptive image. Save your effort as JPEG

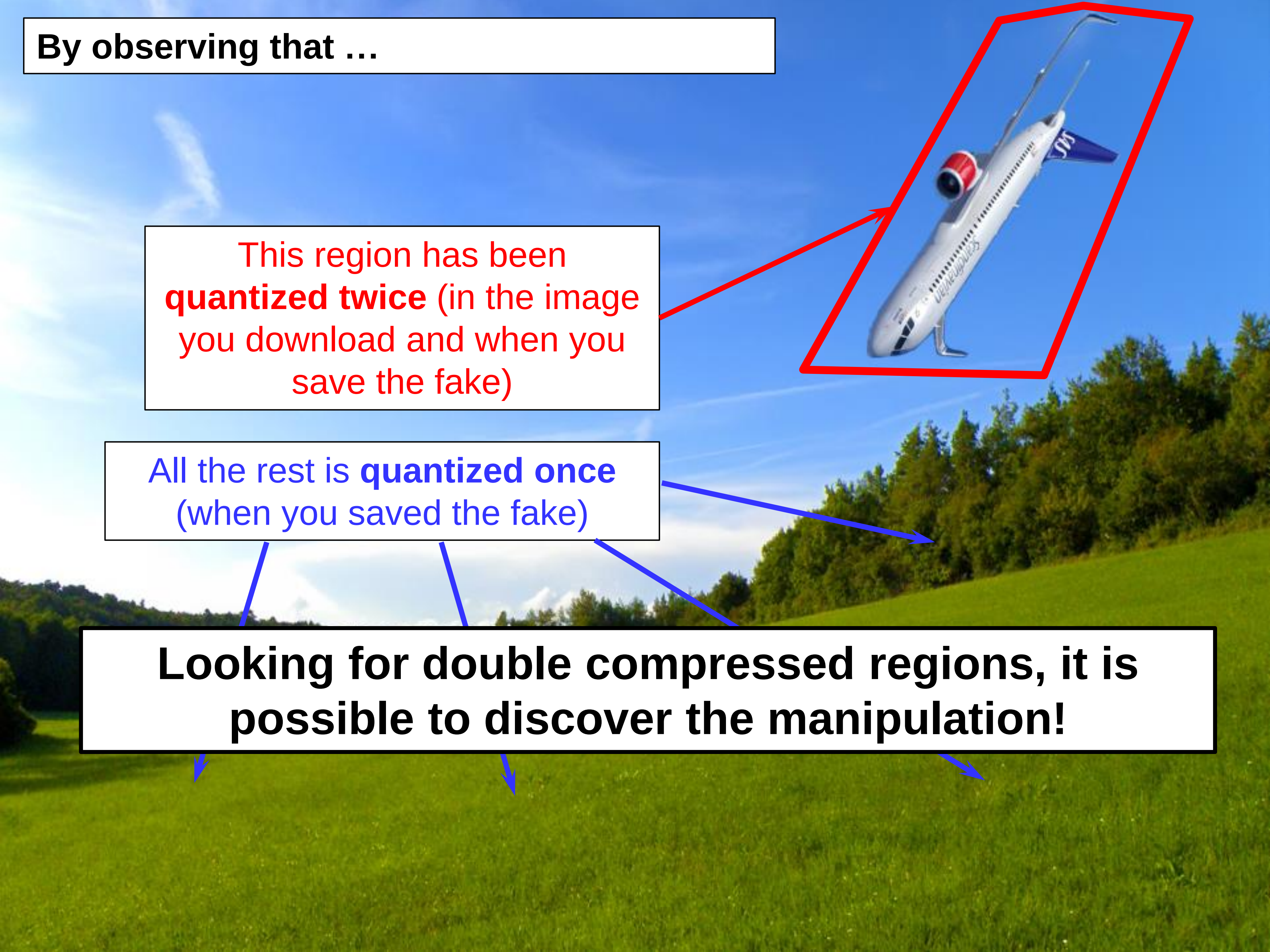
Create a fake, realistic and deceptive image. Save your effort as JPEG

How can one reveal your manipulation?

By observing that …



By observing that …



This region has been **quantized twice** (in the image you download and when you save the fake)

All the rest is **quantized once** (when you saved the fake)

Looking for double compressed regions, it is possible to discover the manipulation!

Double JPEG compression: footprints

Why understanding whether an image has been JPEG compressed (quantized) twice is important?

Double compression is telltale of manipulation

Double quantization: footprints

- When an image is JPEG compressed first with QF_1 and then JPEG compressed again with QF_2, a **double quantization** occurs.
- **Statistical footprints are left by double quantization** !
- Then, double JPEG images show these artifacts (while single JPEG doesn't !).
- D-JPEG detection can be performed based on these artifacts[*]

Why double quantization leaves footprints?......

[*] Popescu, Alin C., and Hany Farid. "Statistical tools for digital forensics."Information *Hiding*. Springer Berlin Heidelberg, 2004.

Single quantization (SQ)

- *Quantization* is the point-wise operation:

$$Q_a(x) = \mathrm{round}\left(\frac{x}{a}\right)$$

- Where:
  - a is a strictly positive integer (quantization step)
  - The value x/a is approximated to the closest integer

- *De-quantization* brings the quantized values back to their original range

$$\hat{x}_a = Q_a(x) \cdot a$$

- **Qa is not invertible because of the rounding operation**

Double quantization (DQ)

- Double quantization is a point-wise operation:

$$Q_{ab}(x) = \text{round}\left(\text{round}\left(\frac{x}{b}\right) \cdot \frac{b}{a}\right)$$

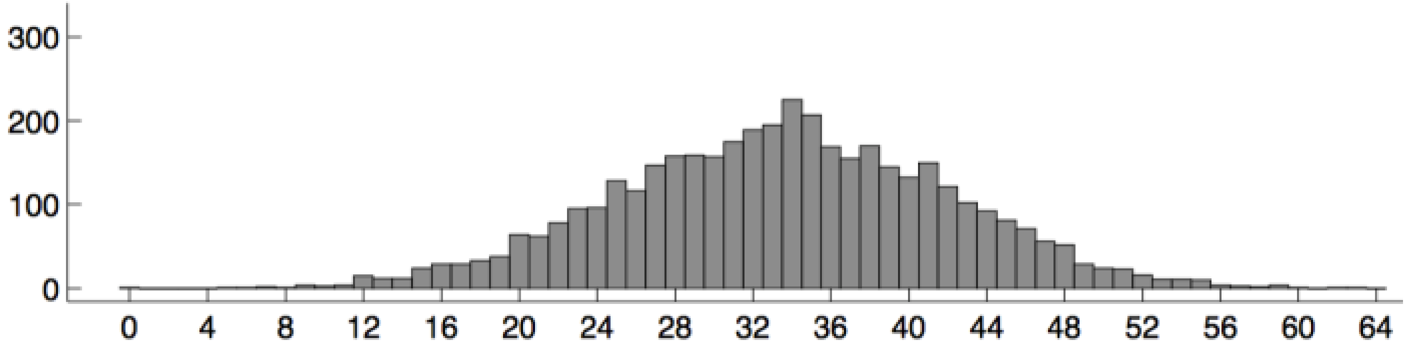
- Where:
  - b and a are the quantization steps of the first and second quantization

- Double quantization can be represented as a sequence of three steps:

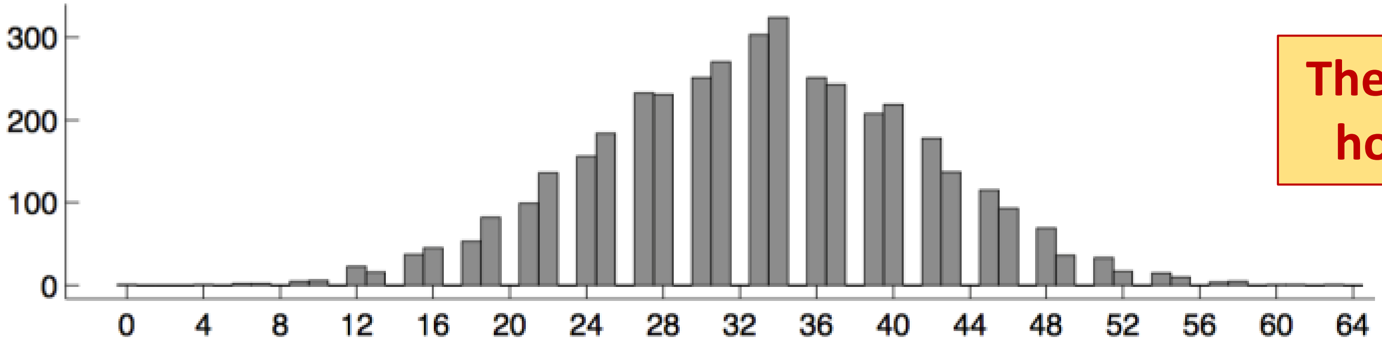
1. Quantization with step b
2. De-quantization with step b
3. Quantization with step a

Double quantization footprints (1/2)

- Consider a signal x whose samples are normally distributed in [0,127].
- The histogram of the signal quantized with step **2** is the following:



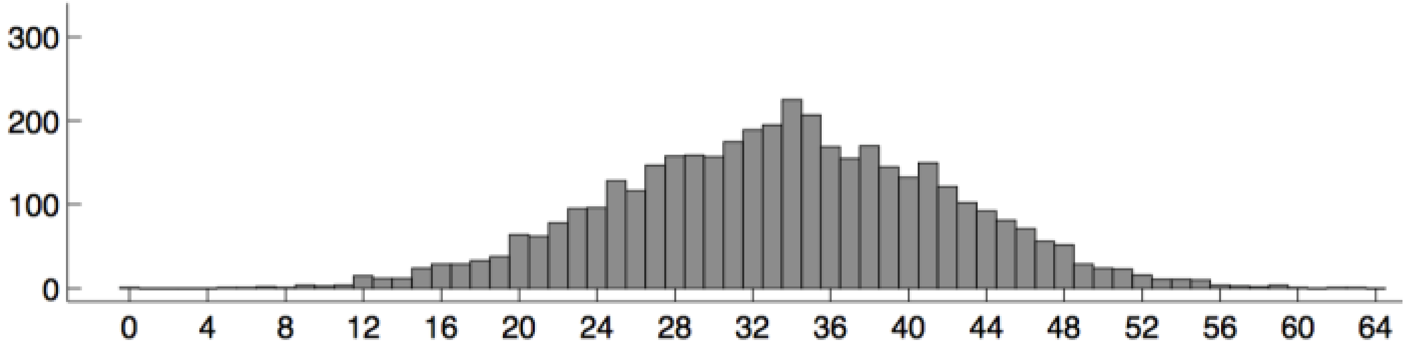
- The histogram of signal quantized with step **3 followed by 2** is :



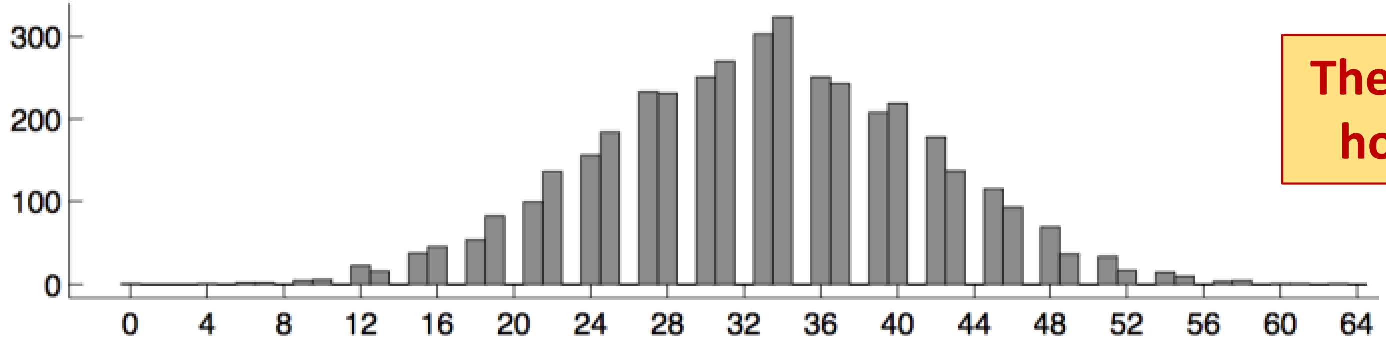
There are holes!!

Double quantization footprints (1/2)

When a<b, some bins are empty (holes). This happens because the second quantization re-distributes the quantized coefficients into more bins than the first quantization



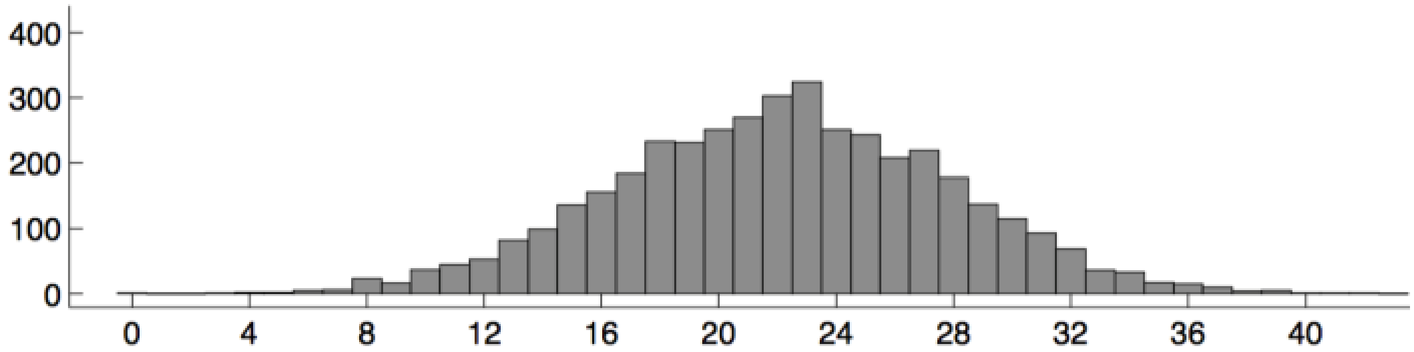
- The histogram of signal quantized with step **3 followed by 2** is :



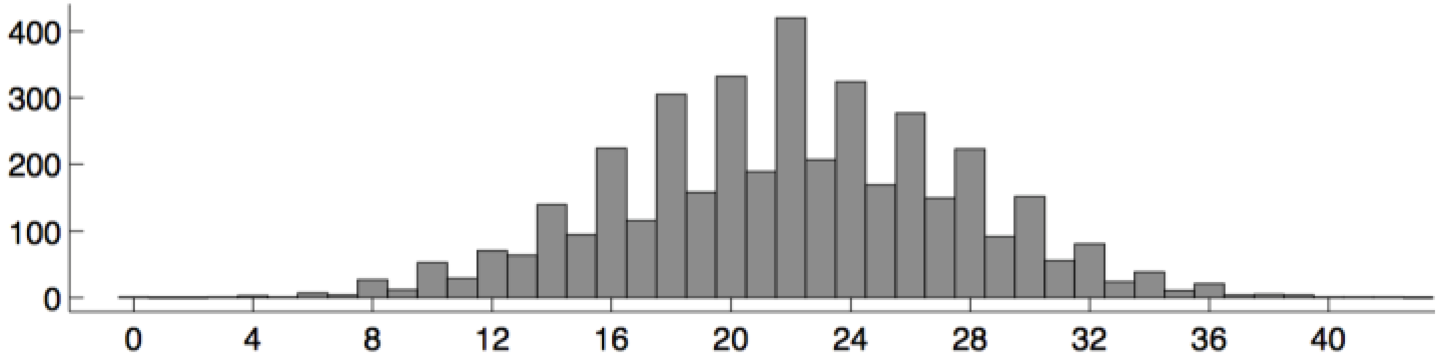
There are holes!!

Double quantization footprints (2/2)

- Consider the same signal, now quantized with step **3**. Its histogram is:

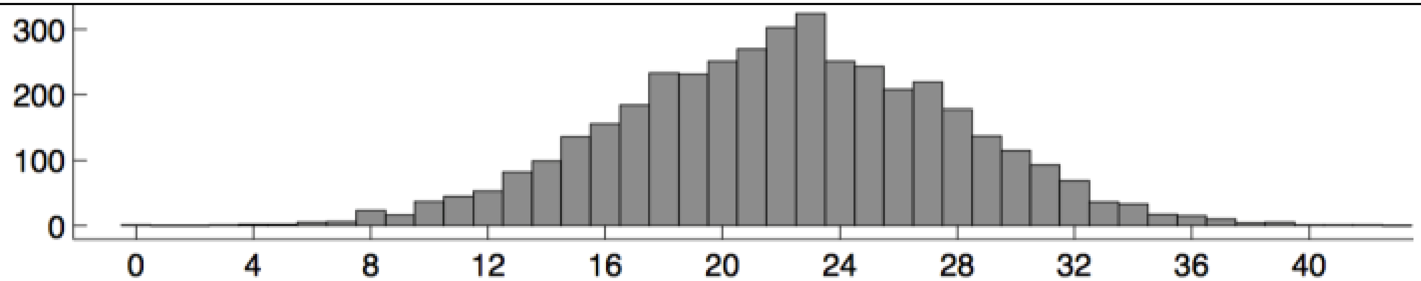


- The histogram of the signal quantized with step **2 followed by 3**:

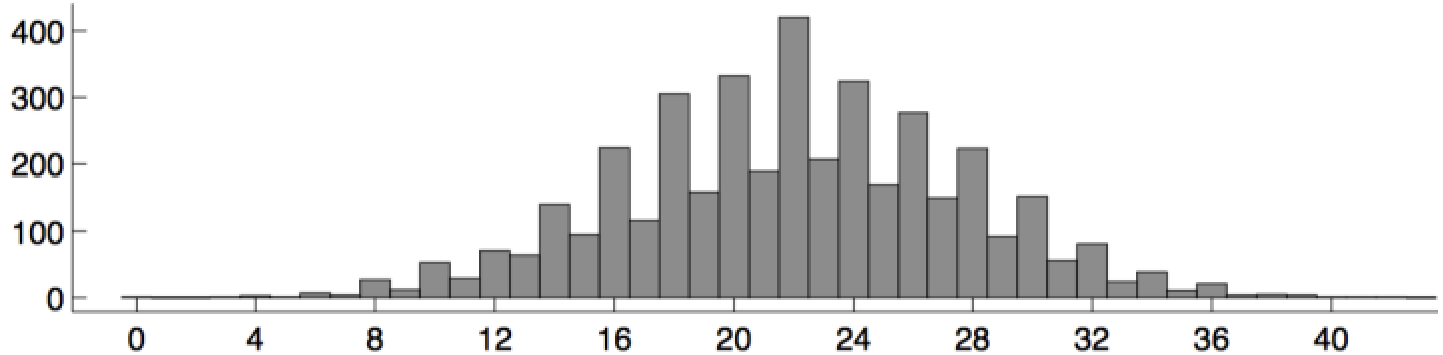


Double quantization footprints (2/2)

When a>b, some bins contain more samples that neighbouring bins. This happens because even bins receive samples from more original bins with respect to the odd bins



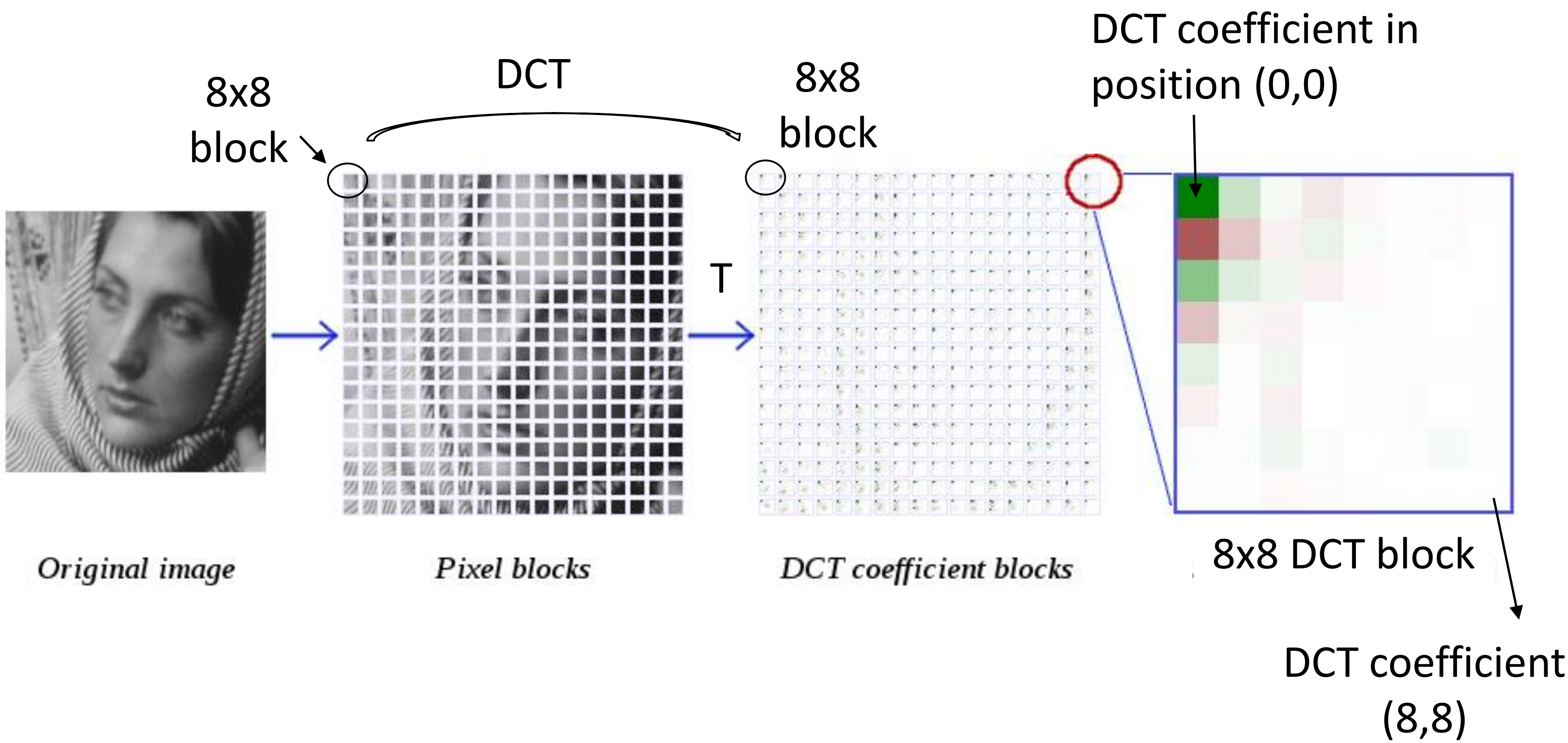
- The histogram of the signal quantized with step **2 followed by 3**:



Double quantization and DJPEG

- In a JPEG image, quantization is performed in the DCT domain
- Then, in a D-JPEG image, the double quantization footprints consist in periodic artifacts in the histograms of the 8x8 block-DCT coefficients
  - When $QF_1 < QF_2$, the histograms have periodic holes

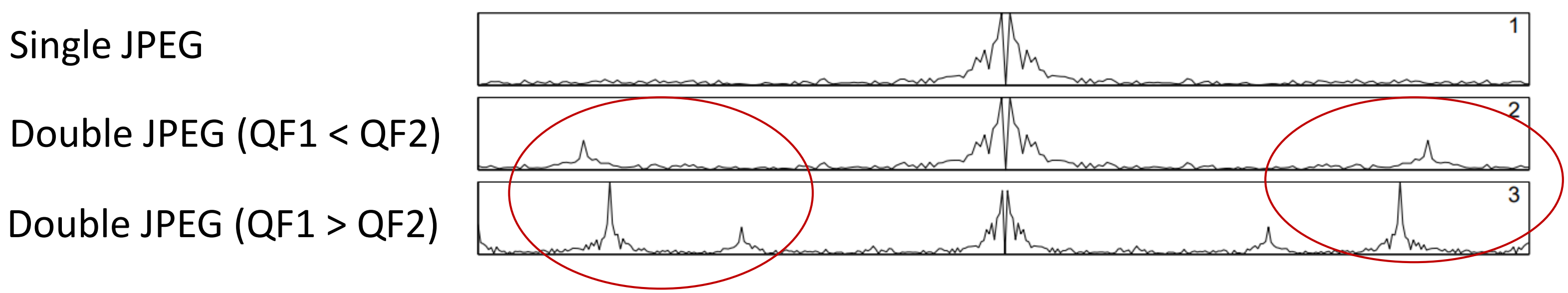
Computing the DCT histograms



- For each of the 64 DCT cofficients, the histogram of the values taken in all the blocks is computed.

Detection of double quantization

- The periodic patterns are particularly visible in the Fourier domain as strong peaks in the mid and high frequencies.
- Then, the Fourier transform of each DCT histogram is evaluated to see if it has certain artifacts [*].
- If the answer is "yes" for at least 1 of the first 10 DCT histograms of the JPEG image, the image is regarded as doubly compressed.
- **Example**: Fourier transform of DCT coeff (1,1)



[*] Popescu, Alin C., and Hany Farid. "Statistical tools for digital forensics."Information *Hiding*. Springer Berlin Heidelberg, 2004.

Detection of double quantization

- **For the case $QF_1 < QF_2$, the detection is more reliable**
  - Peaks and gap are easy to detect….There are *holes*!
  - *Rule of thumb:*

$$\Delta QF = QF_2 - QF_1 \geq 10$$

(the strength of the artifacts depends on Δ)

- $QF_1 < QF_2$ is often the most frequent case in practice

[*] Popescu, Alin C., and Hany Farid. "Statistical tools for digital forensics."Information *Hiding*. Springer Berlin Heidelberg, 2004.

Detection of double JPEG compression

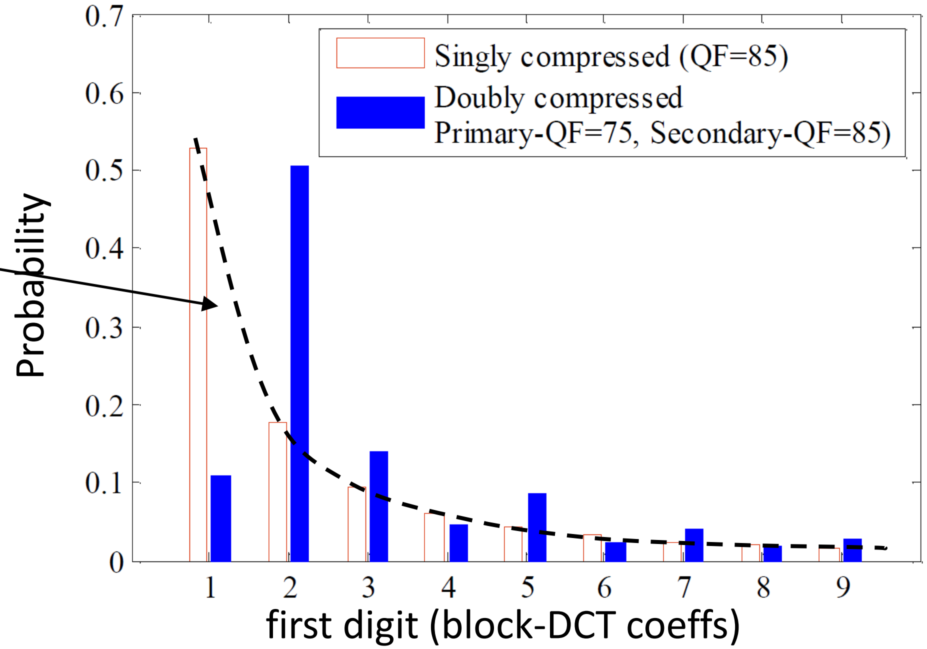
- Several detectors of double JPEG compression proposed in Image Forensics

1. Popescu, Alin C., and Hany Farid. "**Statistical tools for digital forensics**."*Information Hiding*. Springer Berlin Heidelberg, 2004.
2. Huang, Fangjun, Jiwu Huang, and Yun Qing Shi. "**Detecting double JPEG compression with the same quantization matrix**." *Information Forensics and Security, IEEE Transactions on* 5.4 (2010): 848-856.
3. Bianchi, Tiziano, and Alessandro Piva. "**Detection of nonaligned double JPEG compression based on integer periodicity maps**." *Information Forensics and Security, IEEE Transactions on* 7.2 (2012): 842-848.
4. Pevný, Tomáš, and Jessica Fridrich. "**Detection of double-compression in JPEG images for applications in steganography**." *Information Forensics and Security, IEEE Transactions on* 3.2 (2008): 247-258.
5. Bianchi, Tiziano, and Alessandro Piva. "**Detection of non-aligned double JPEG compression with estimation of primary compression parameters**." *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011.
6. Lukáš, Jan, and Jessica Fridrich. "**Estimation of primary quantization matrix in double compressed JPEG images**." *Proc. Digital Forensic Research Workshop*. 2003.
7. Fu, Dongdong, Yun Q. Shi, and Wei Su. "**A generalized Benford's law for JPEG coefficients and its applications in image forensics**." *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007.
8. He, Junfeng, et al. "**Detecting doctored JPEG images via DCT coefficient analysis**." *Computer Vision–ECCV 2006*. Springer Berlin Heidelberg, 2006. 423-435.

Another feature: FSD distribution

- Another method looks at the **distribution of the First Significant Digits (FSD)** of the block-DCT coefficients.
- For single JPEG images, the distribution of the FSDs follows a known law (*Benford's law*) [**]
- Double compression cause violation of this law
- **Example:**

$$p_i = \log_{10}\left(1 + \frac{1}{i}\right)$$



[**] Fu, Dongdong, Yun Q. Shi, and Wei Su. "A generalized Benford's law for JPEG coefficients and its applications in image forensics." Electronic Imaging 2007. International Society for Optics and Photonics, 2007.

Beyond model-based approaches

- We have seen examples of *model-based* approaches (relying on statistical models)
- Another category of (more powerful) methods: *Data-driven* approaches
- **What is Data-driven (or Machine Learning-based) classification ?**

Data-driven (machine-learning based) classification

Why machine learning?

- **Probabilistic models are often unknown** (in real application scenarios)
- A statistical characterization may even not be possible.
- Then, model-based approaches for data analysis are not viable (possible only under particular conditions)
- …we need to resort to machine learning approaches!!
- **Machine Learning (ML) is about learning structure from data, namely ‘examples’.**
  - E.g., in a binary classification problem: the statistical characterization of a given phenomenon under H0 and H1 is unknown…but samples from the two classes are available !

An example (binary classification)

- Suppose we have 50 photographs/images of elephants (H0) and 50 photos of tigers (H1).

vs

- Now, given a new (different) photograph/image we want to answer the question: is it an elephant or a tiger? [assuming that it is either one or the other.]

An example (binary classification)

- Suppos ephants (H0) and 50

- Now, given a new (different) photograph/image we want to answer the question: is it an elephant or a tiger? [assuming that it is either one or the other.]

Formally…

- We want the system *to learn* the mapping: X → Y, where **x** ∈ X is some *object (feature vector)* and y ∈ Y is a *class label*.
- Simplest case: 2-class classification: x ∈ R^n, y ∈ {±1}.
- **Training set** (made of labeled examples): $(x^1,\ y^1)$,..., $(x^m,\ y^m)$
- Generalization purpose: given a previously unseen **x** ∈ X , determine y ∈ Y
- **ML methods learn a classification function** y = f (**x**, α), **for a given** f, **where α is a set of unknown parameters of the function, to be optimized.**
- **These unknown parameters are optimized ("learned") on the training set.**

ML algorithms

- **Support Vector Machines (SVM) or Networks**
  - The simplest ML algorithm (one of the most commonly used) for classification and estimation problems
- **Neural Networks (NN)**
- These networks are usually fed with *feature vectors* (x ∈ R^n is a feature vector).

The recent trend:

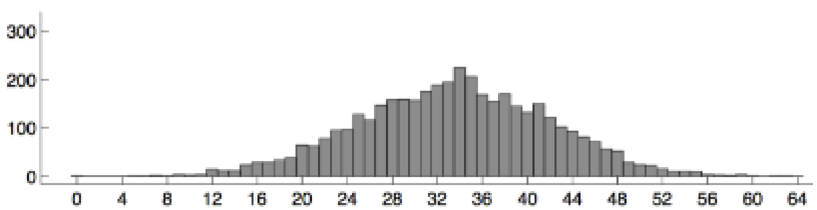
- **Deep Neural Networks (DNN), and Convolutional Neural Network (CNN)**
  - Outstanding performance
  - x ∈ R^n can be an image (image block). The features are *self-learned* by the CNN.

SVM-based double JPEG detection

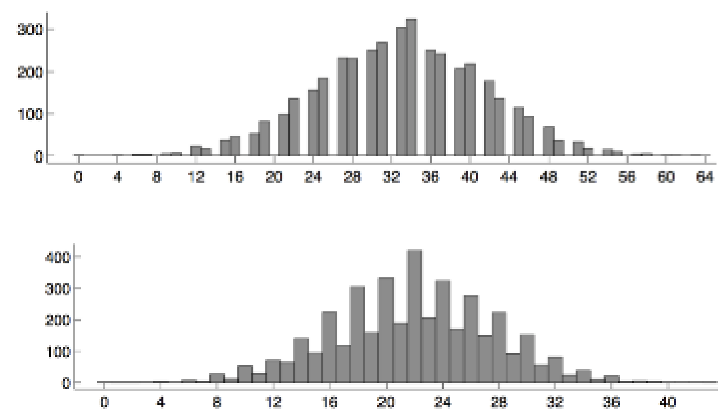
- We can use machine learning techniques to build a classifier that can distinguish between *single JPEG* images (H0) and *double JPEG* images (H1)…..
- Several approaches have been proposed

SVM-based double JPEG detection

- Through SVMs, we can build a detector that can distinguish between single quantized DCT histograms ("without artifacts") and double quantized DCT histograms (with "artifacts")…..



H0

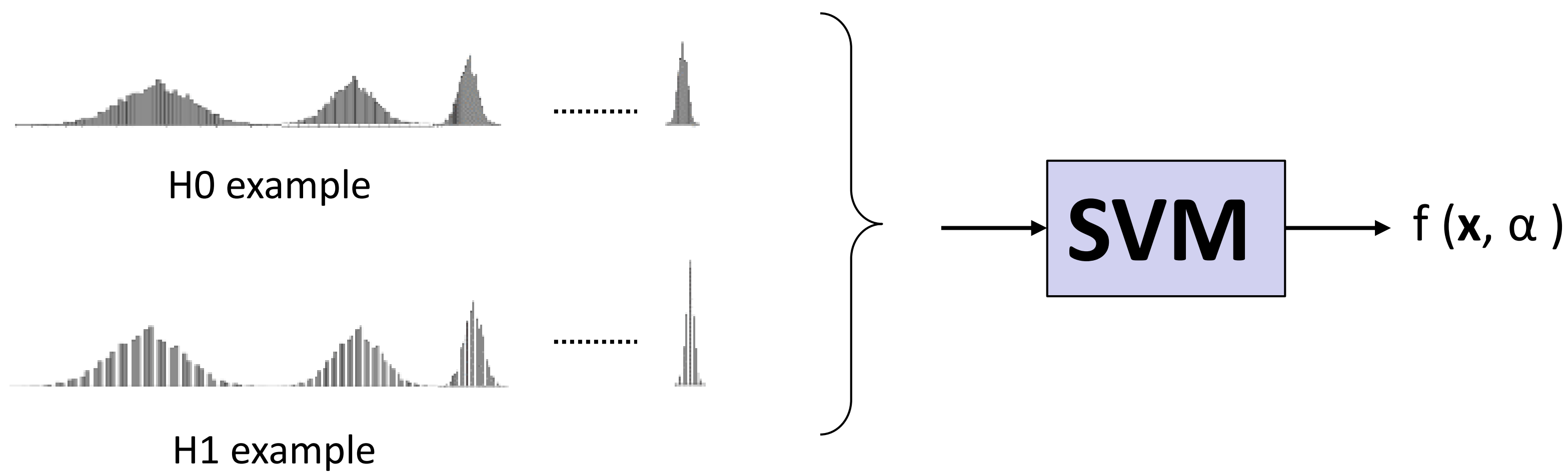


H1

Very Easy!!

SVM-based double JPEG detection

- The histograms of the 64 block-DCT coefficients can be concatenated (forming a feature vector) [***]
- This feature vector can be given as input to an SVM classifier…
- **Example** (of input feature vector **x**):



[***] Pevný, Tomáš, and Jessica Fridrich. "Detection of double-compression in JPEG images for applications in steganography." Information Forensics and Security, IEEE Transactions on 3.2 (2008): 247-258

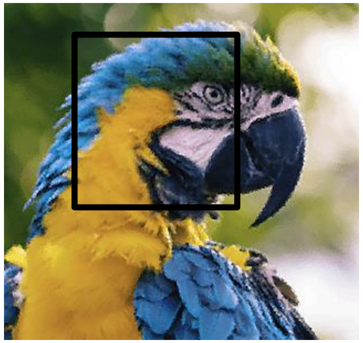
Rich feature sets

- General **rich sets of features** have been derived [#], computed from either *the DCT image* and *the pixel image* (first and higher-order features)
- This rich sets of features can be used to train SVM (or NN) models to address several classification task (not only DJPEG !)
  - traces can be captured either by the frequency (DCT) domain features or the pixel domain features
- For D-JPEG detection, even better performance can be obtained (especially in the most difficult cases, e.g., QF1 ≈ QF2)

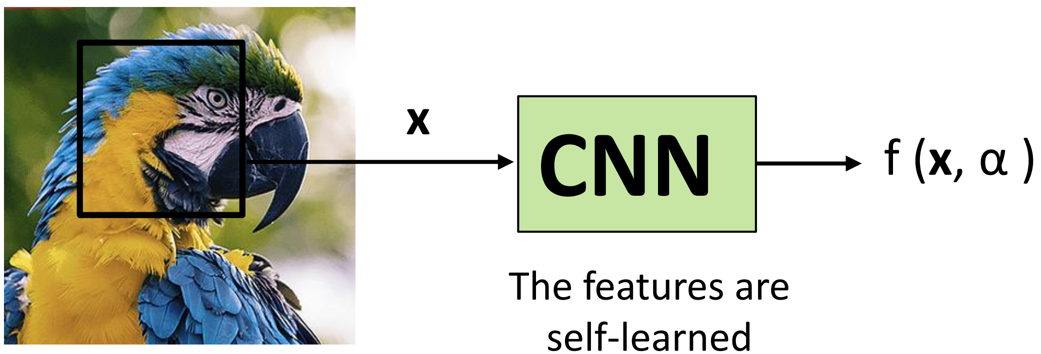
[#] Jessica Fridrich and Jan Kodovský. "Rich Models for Steganalysis of Digital Images." IEEE Transactions on Information Forensics and Security, 7(3), 868-882

CNN-based DJPEG detection

- With the adoption of CNN models, it is possible to boost the performance of D-JPEG detection [&]
- A CNN model can be successfully trained, *directly* from the image (or image regions)
- A large amount of traininig data are necessary (representative for all the cases of (QF1,QF2))



H1 example



H0 example

The features are self-learned

[&] Barni, M., Bondi, L., Bonettini, N., Bestagini, P., Costanzo, A., Maggini, M., Tondi, B., Tubaro, S. (2017). Aligned and non-aligned double JPEG detection using convolutional neural networks. Journal of Visual Communication and Image Representation, 49, 153-163.

Data-driven (Machine Learning-based) vs Model-based

Data-driven vs Model-based approaches

- **Strengths of D-D methods:**
  - *Much better performance in general*
  - *Capable to work under very general conditions.* For Double JPEG detection, a D-D method could work for:
    - QF1 > or < than QF2
    - Aigned or not aligned JPEG (the artifacts are different in the aligned and misaligned case)
  - *Capable to work in difficult cases* (QF1 ≈ QF2, that is, ΔQF is small)

Data-driven vs Model-based appraoches

- **Weakness of DD methods:**
  - *Are the ''learned'' features are (really) peculiar for the detection task under consideration ?*
    - DD solution may rely on (so called) *confounding factor*...
  - *Huge amount of data required* (**big-data problem**)
  - *The performance decrease on different image datasets* (**dataset mismatch problem**)
    - Sensitiveness to image properties (e.g., resolution,..)
  - *Then, the training phase is very critical !*