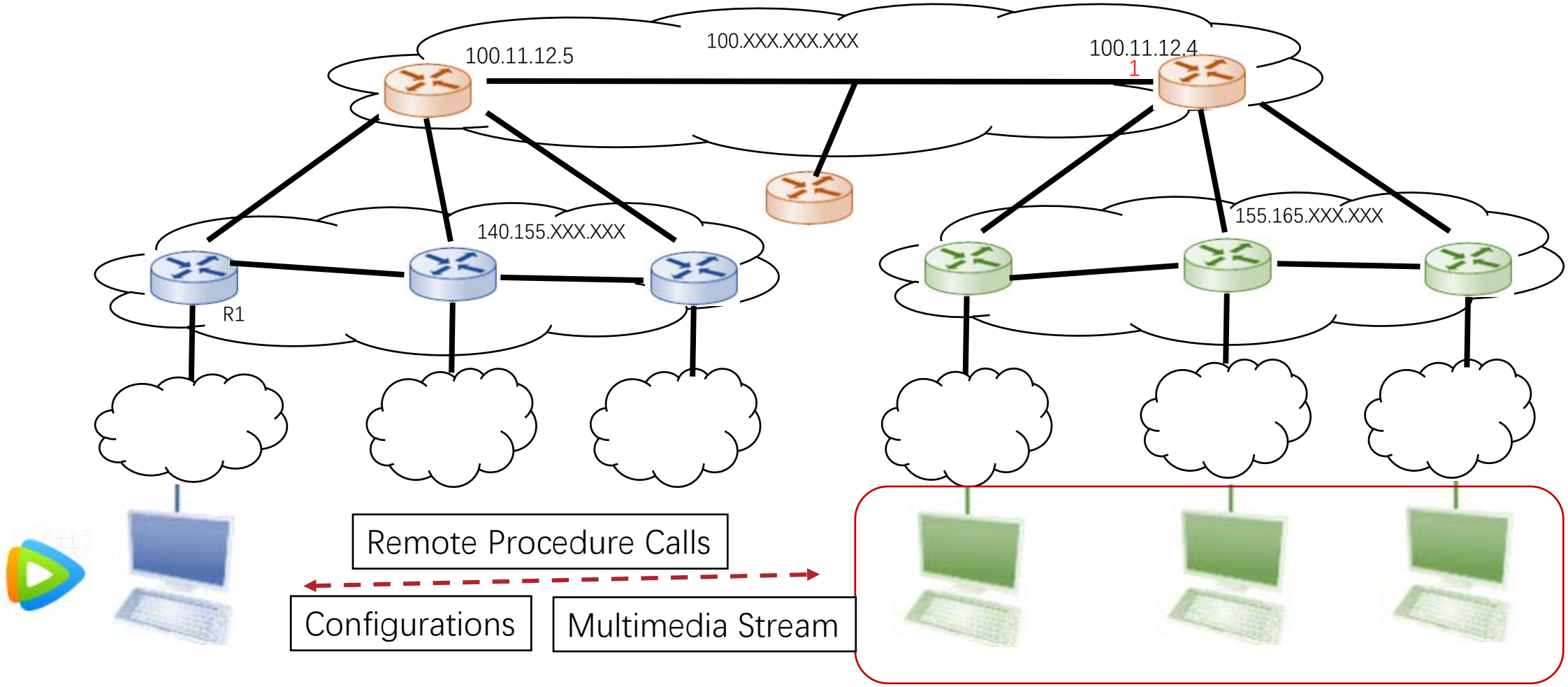
CS120: Computer Networks

Lecture 23. Data Compression

Zhice Yang

Data in End-to-End Connections



Data in End-to-End Connections

- Data Presentation
- ➢Data Compression
  - Lossless Compression
  - Multimedia Compression

gzip

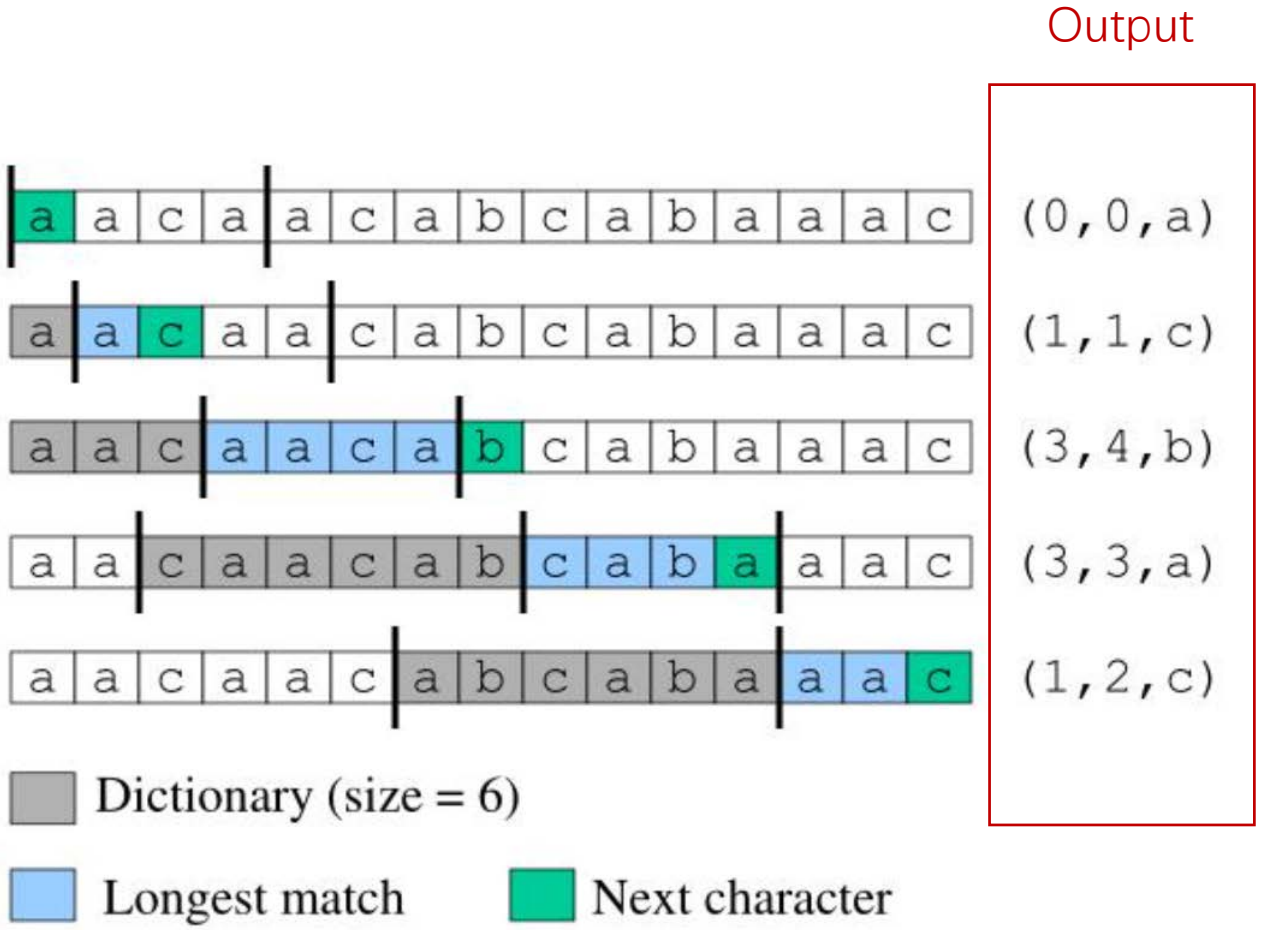
- GNU zip
- A widely-used lossless compression method
- Main Algorithms
  - LZ Algorithm
  - Huffman Coding

LZ Algorithm

- Dictionary-Based Compression
- Method
  - Construct dictionary: find repeated strings
  - Repeated strings are represented by its index in dictionary
    - eg. Repeated strings are simplified to <distance, length> pair
      - blah blah b! => blah [D=5, L=6]!

LZ77

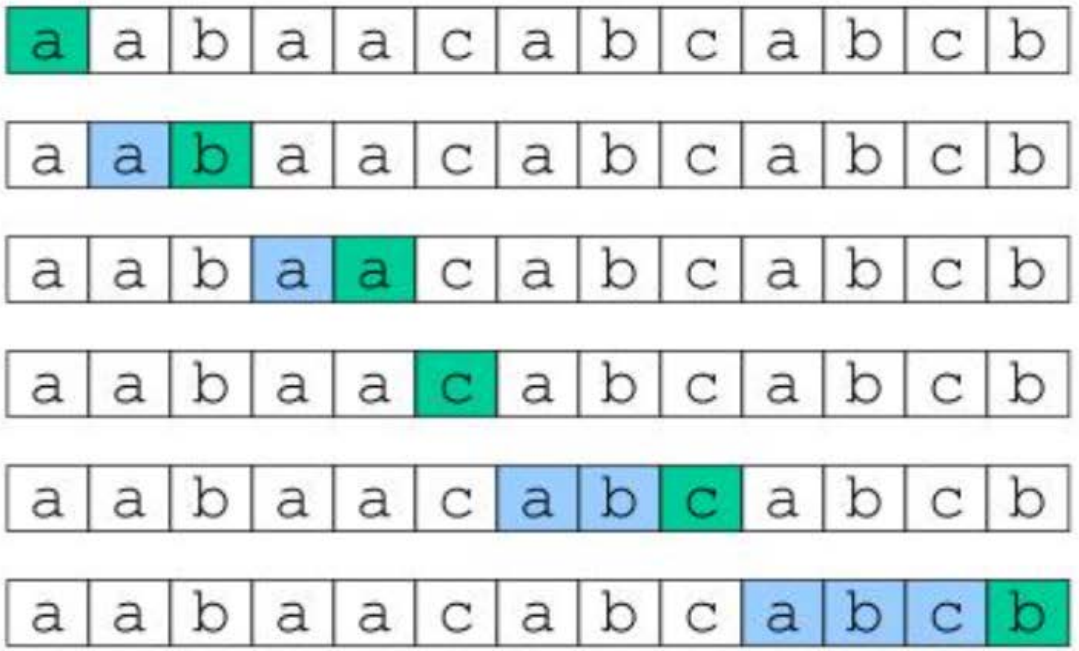
- Encoding



https://www.slideserve.com/shalin/data-compression

LZ78

- Encoding



Output

Output	Dict.
(0,a)	1 = a
(1,b)	2 = ab
(1,a)	3 = aa
(0,c)	4 = c
(2,c)	5 = abc
(5,b)	6 = abcb

https://www.slideserve.com/shalin/data-compression

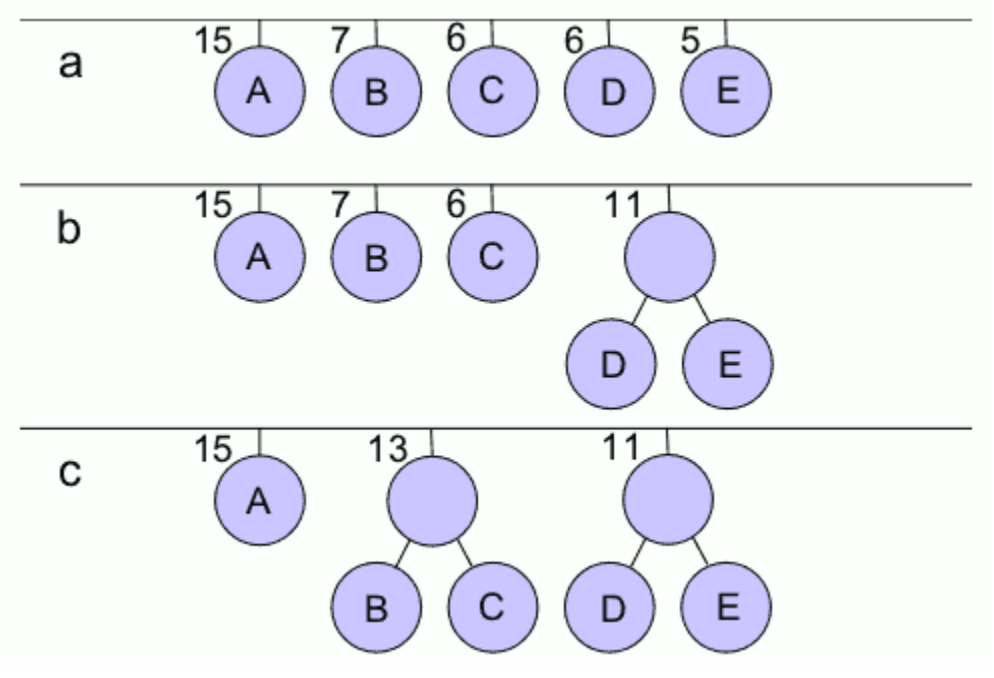
Huffman Coding

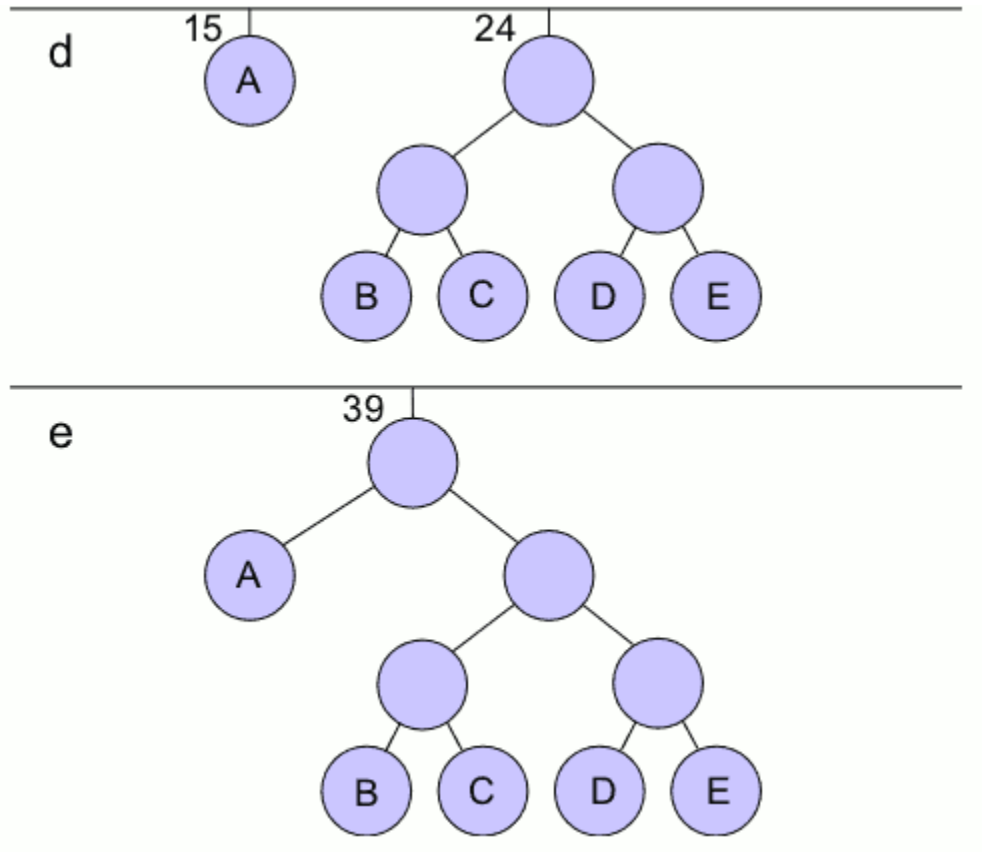
- Intuition: Higher frequency characters => less bit to representation
- A:90%, B:5%, C:5%
  - A: 1
  - B: 01
  - C: 00

Huffman Coding

- Create a leaf node for each symbol and add it to the priority queue.
- While there is more than one node in the queue:
  - Remove the two nodes of highest priority (lowest probability) from the queue
  - Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
  - Add the new node to the queue.
- The remaining node is the root node and the tree is complete.

Huffman Coding





ref: https://en.wikipedia.org/wiki/Huffman_coding

Data in End-to-End Connections

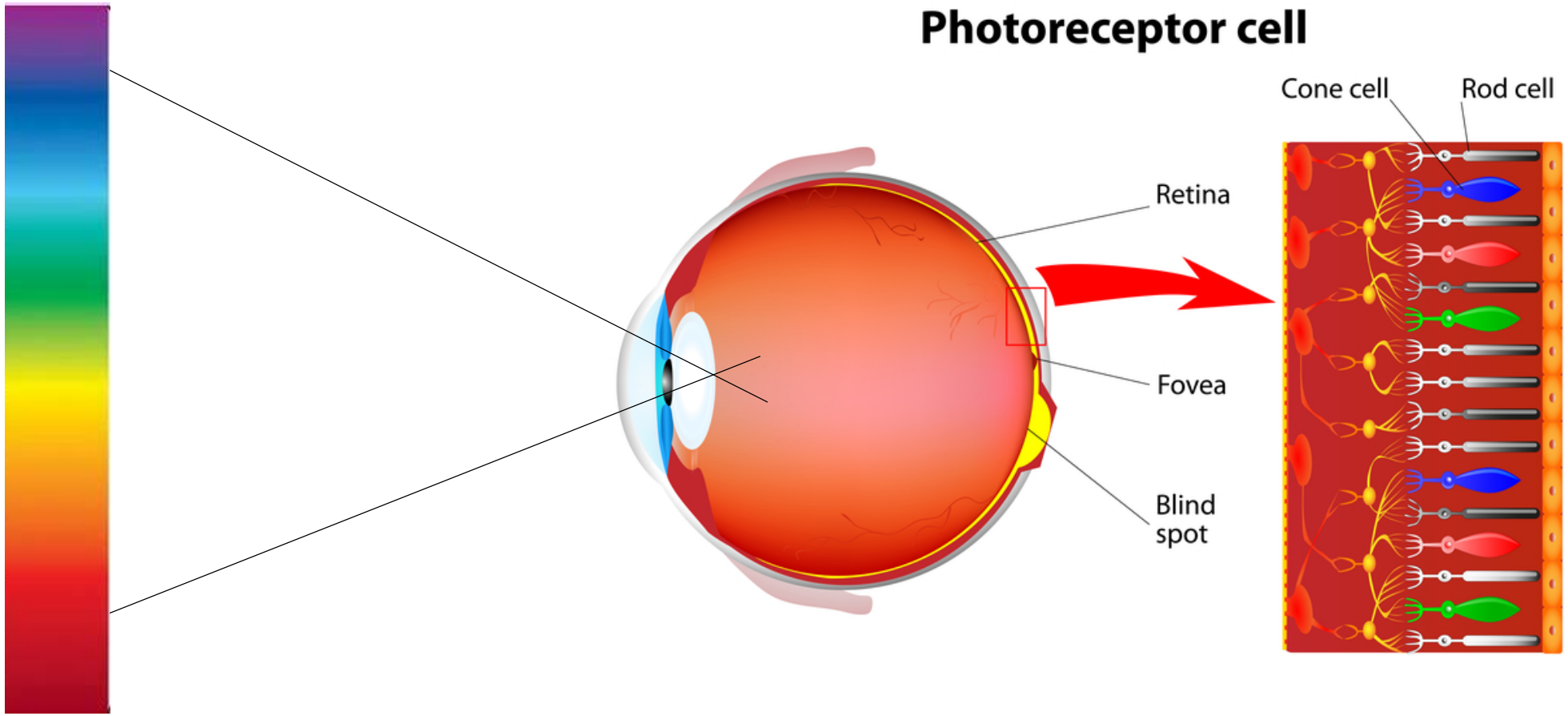
- Data Presentation
- Data Compression
  - Lossless Compression
  - ➢Multimedia Compression

End-to-End Data

- Data Presentation
- Data Compression
  - Lossless Compression
  - ➢Multimedia Compression

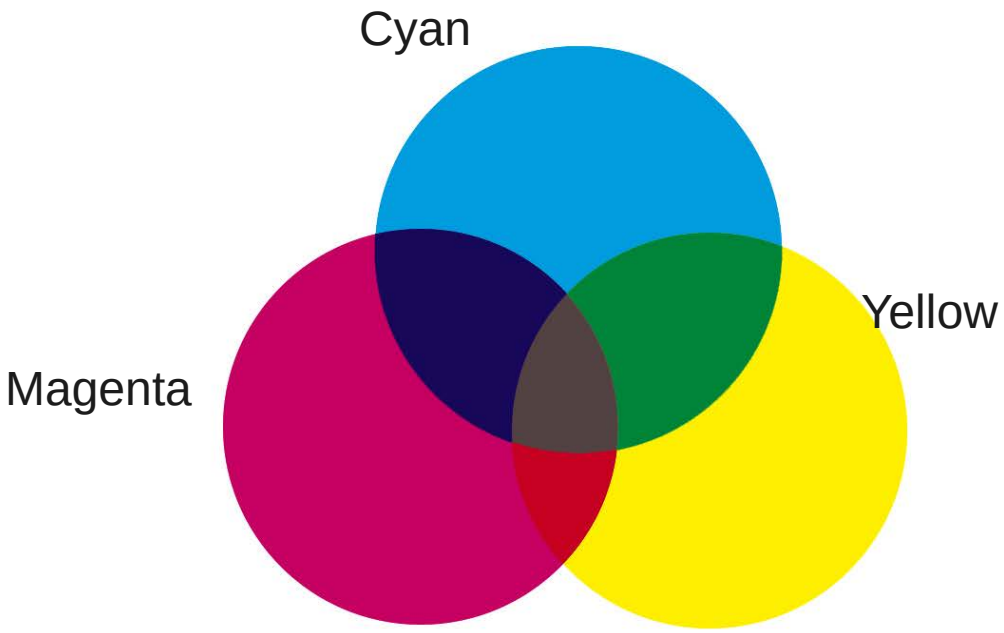
Color and Display

- eyes spectrum



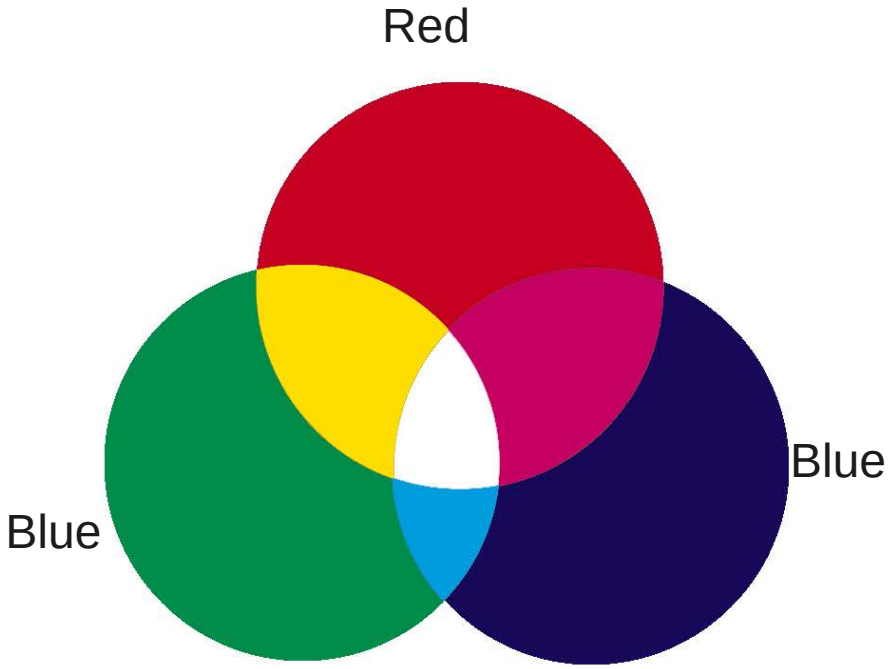
Color and Display

- Color Model



Subtractive color (CMYK)

for printing

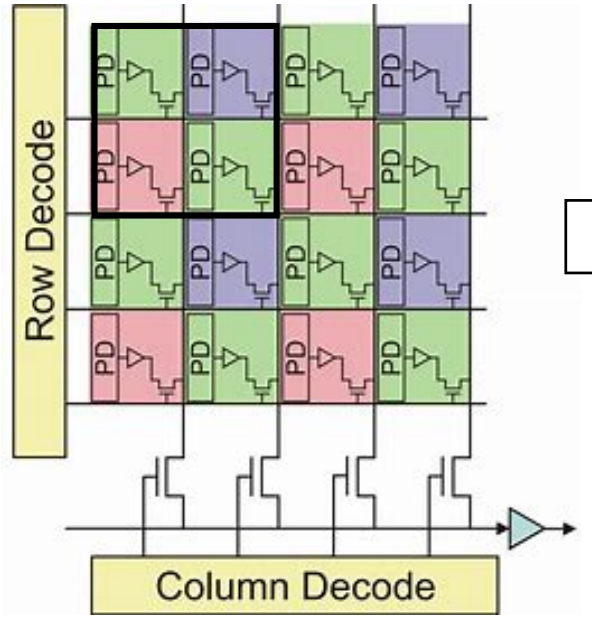


Additive Color (RGB)

for display

Color and Display

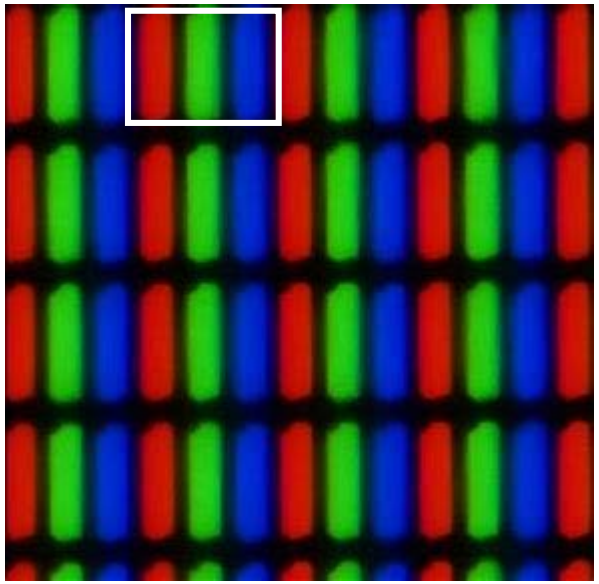
- Imaging and Display



CMOS

RGBs

RGBs



Display

Color and Display

- Digital Image

$$= \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a & & \cdots & \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & & & \\ \vdots & & & \\ a_{n1} & & & \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}_{n \times m}$$

Size = 1080*1920*8*3 = 50Mb !

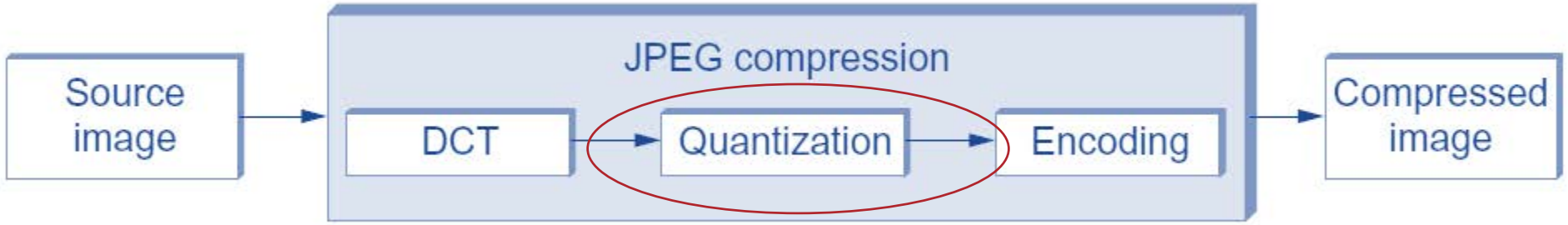
GIF – Image Compression

- Filename Extension: **`.gif`**
- Simple Lossy Compression
- 3*8 bit => 256 colors

JPEG – Image Compression

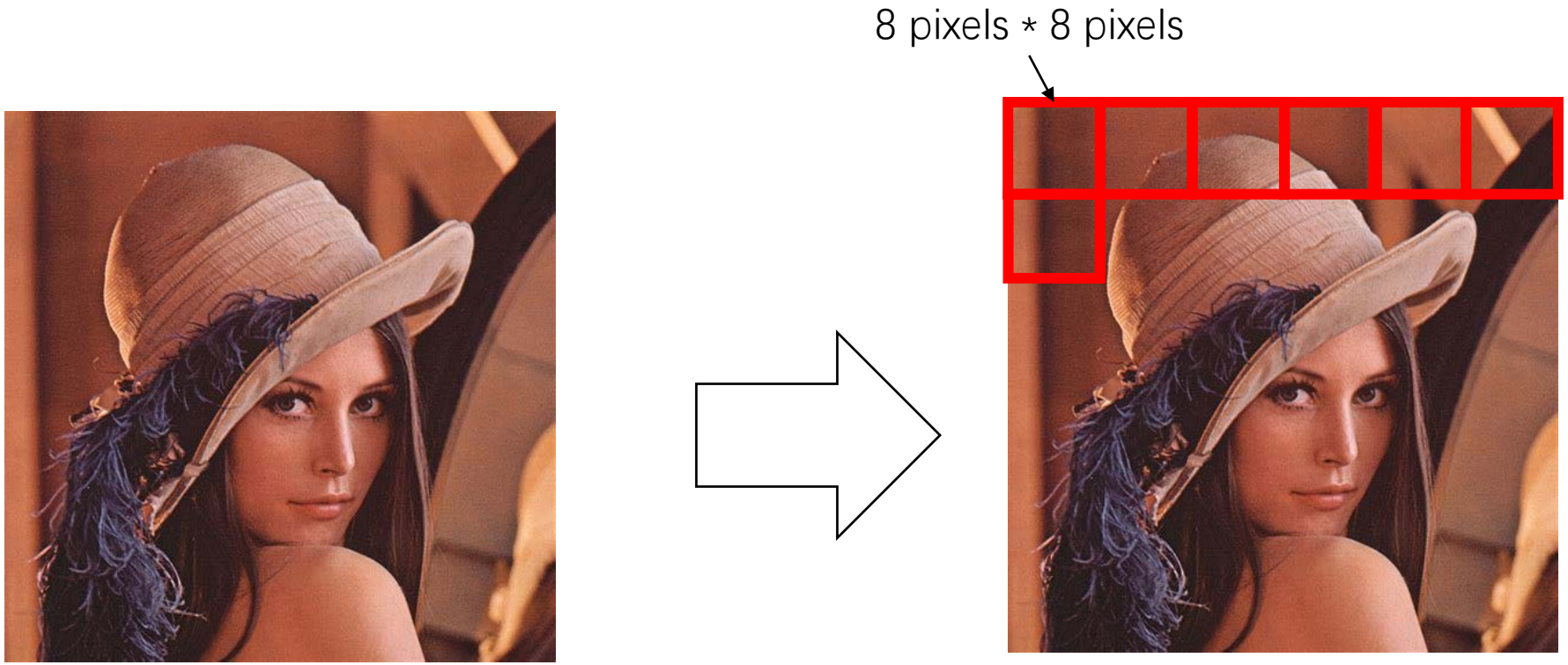
- Filename Extension: `.jpg, .jpeg`
- Joint Photographic Experts Group
- Intuition
  - Human eyes are sensitive to intensity changes, but less sensitive to chromatic changes
  - Human eyes are sensitive to low frequency changes, but less sensitive to high frequency changes

JPEG Compression Flow



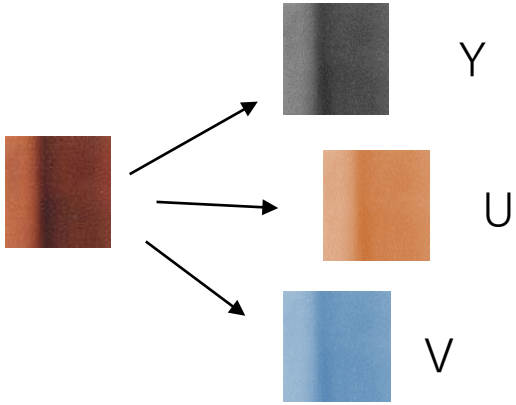
Information Loss

JPEG Compression: Splitting



JPEG Compression: RGB -> YUV

- YUV Space
  - Y -> luminance
    - Sensitive
  - U, V -> chrominance



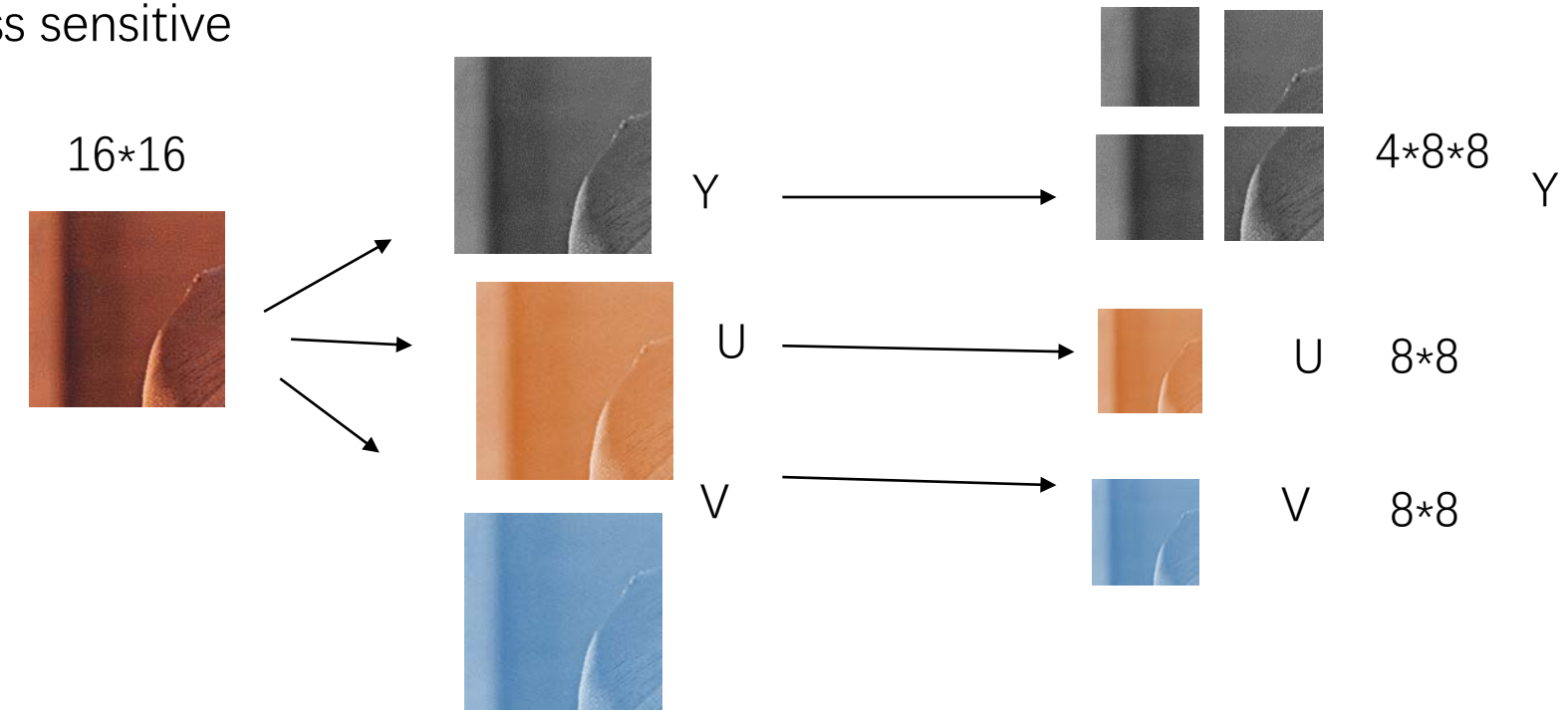
$$Y = 0.299R + 0.587G + 0.114B$$

$$U = (B - Y) \times 0.565$$

$$V = (R - Y) \times 0.713$$

JPEG Compression: Subsampling UV

- YUV Space
  - Y -> luminance
    - Sensitive
  - U, V -> chrominance
    - Less sensitive

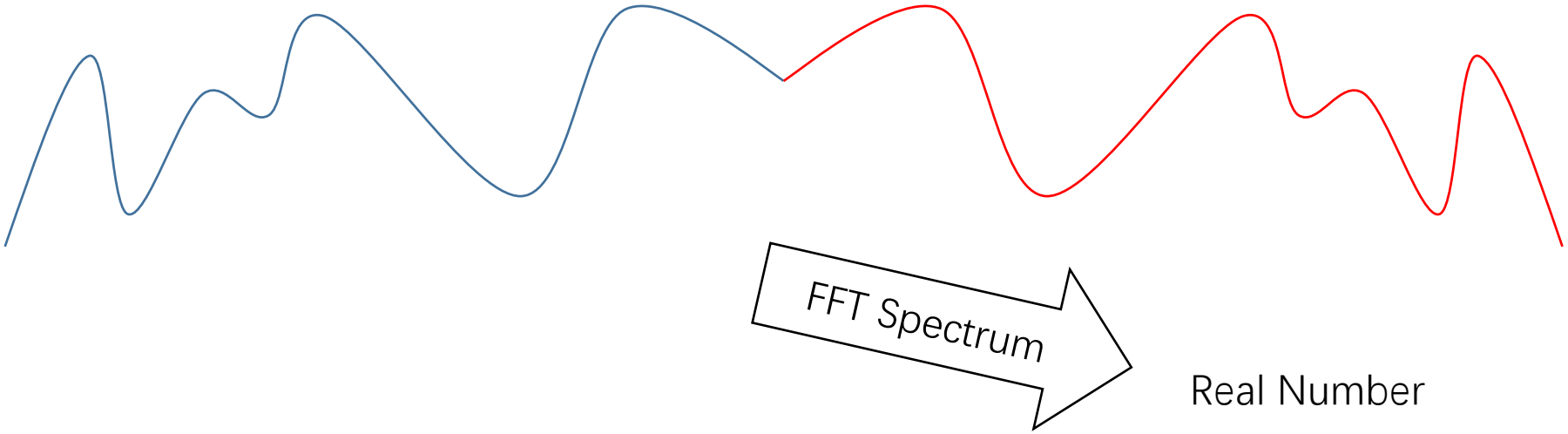


JPEG Compression: DCT

- Discrete Cosine Transform

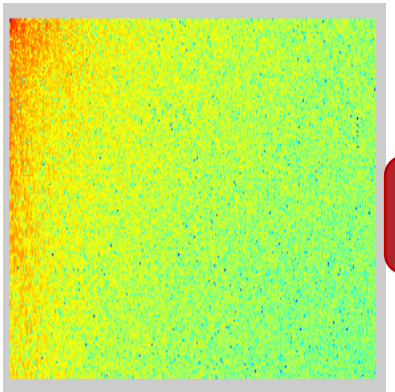
JPEG Compression: DCT

- Discrete Cosine Transform

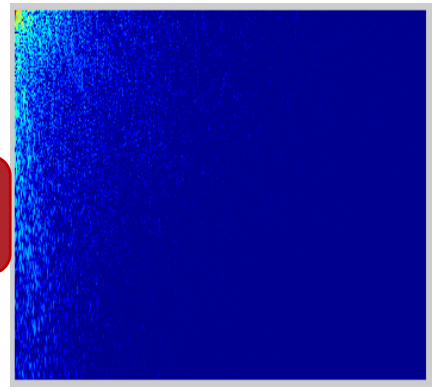


JPEG Compression: DCT

DCT

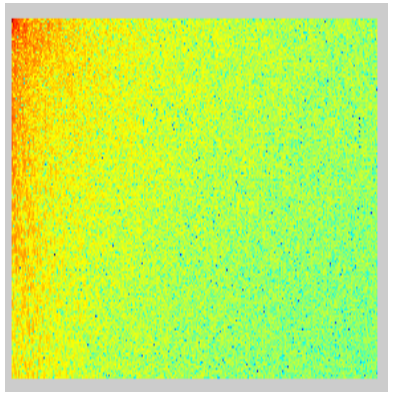


Sensitive to Low frequency

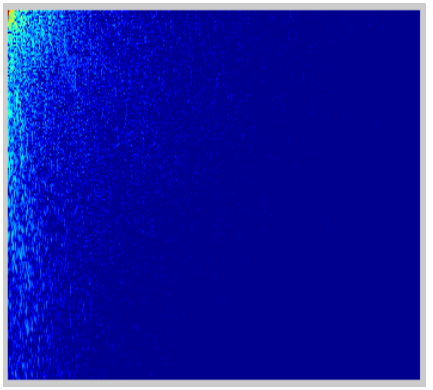


DCT

JPEG Compression: Quantization



Round (DCT(i,j)/Quantum(i,j))



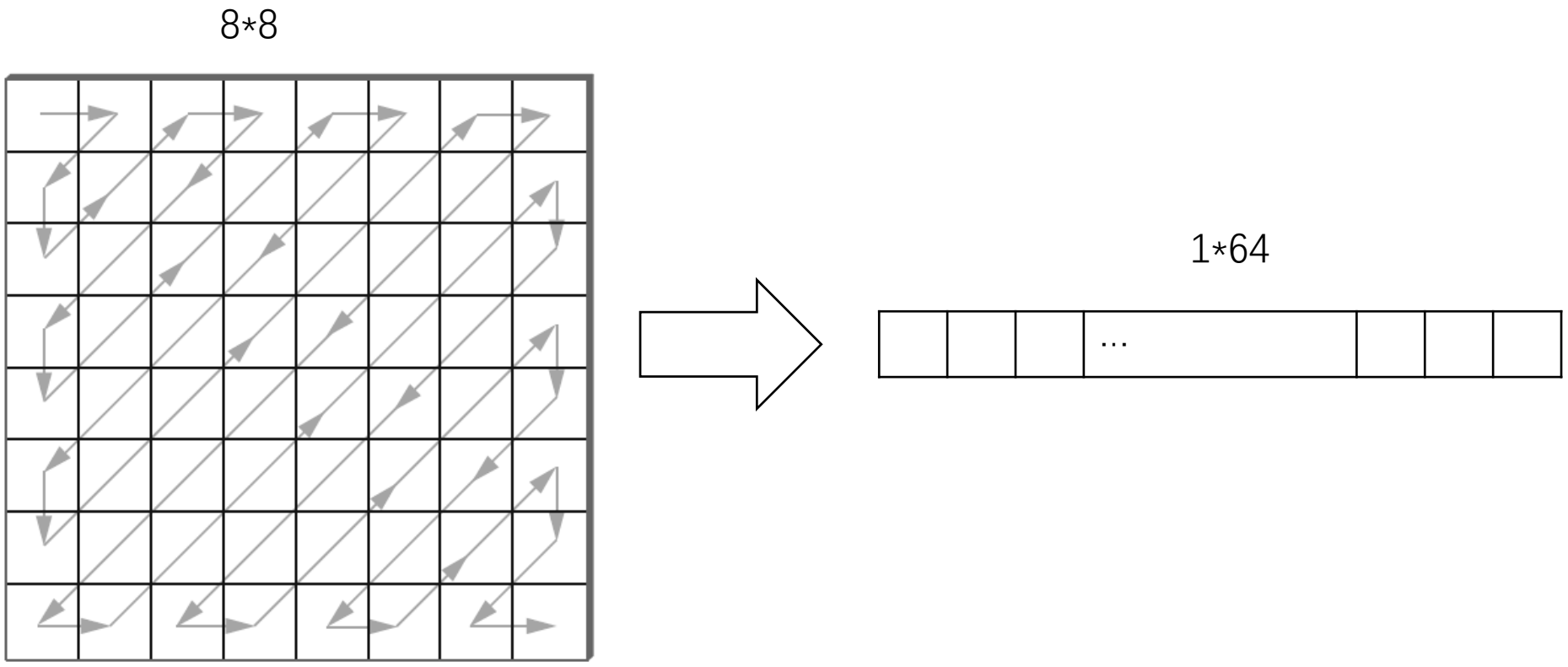
$$\text{Quantum} = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix}$$

Determine the How Much Information is dropped

JPEG Compression: Quantization

- Quantization is a lossy process
  - Recovered DCT(i,j) = QuantizedValue(i,j)*Quantum(i,j)
  - Rounding in Quantization is lossy

JPEG Compression: Zig-Zag

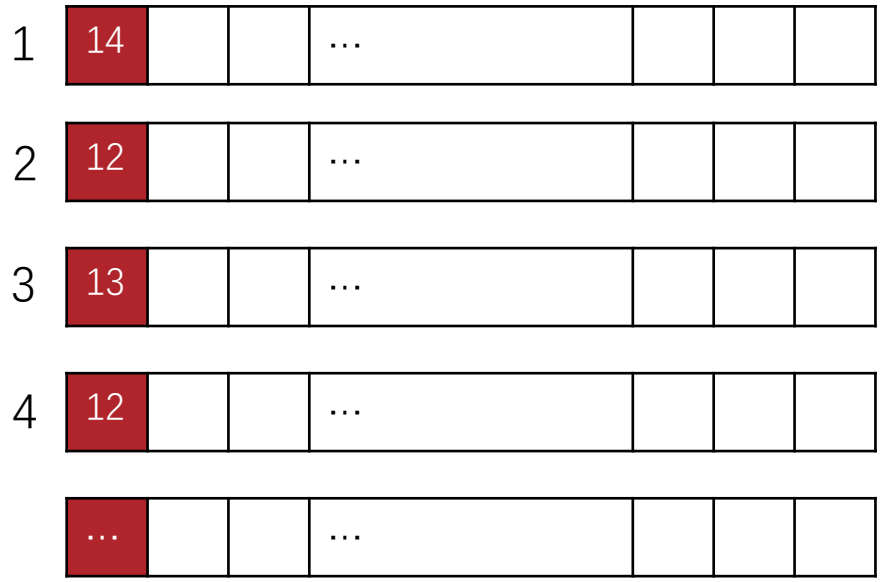


JPEG Compression: DC Component

- DC Components are large and normally non-zero
- Nearby DC Components are closed

➢Differential Pulse Code Modulation (DPCM)

- 14, 12, 13, 12, 15 => 14, -2, 1, -1, 3



JPEG Compression: DC Component

- DC Component can be expressed in integer
  - eg. in one's complement
    - 3 => 0011
    - -3 => 1100
    - 4 => 0100
    - -4 => 1011
- Problem
  - If expressing integer in fix-length bits
    - padding zeros waste space
  - If expressing integer in dynamic length bits
    - how to split the bit stream ?

JPEG Compression

- DC Component can be expressed as (size, amplitude)
  - Size: number of bits to express amplitude
  - Amplitude: DPCM value in ones complement
  - Examples:
    - 0 => (0,-)
    - 1 => (1,1)
    - -1 => (1,0) bitwise inverse for negative value
    - 2 => (2, 10)
    - -2 => (2, 01)
    - 3 => (2,11)
    - -3 => (2,00)

JPEG Compression: Huffman Coding

- DC Component can be expressed as (size, amplitude)
  - Size: number of bits to express amplitude, Huffman coded
    - The coding table is included in the JPEG file
  - Amplitude: DPCM value in ones complement
  - Examples:
    - 0 => (0,-) => 0
    - 1 => (1,1) => 101 1
    - -1 => (1,0) => 101 0
    - 2 => (2, 10) => 011 10
    - -2 => (2, 01) => 011 01
    - 3 => (2,11) => 011 11
    - -3 => (2,00) => 011 00

Length	Code	Size
3 bits	000 001 010 011 100 101 110	04 05 03 02 06 01 00 (End of Block)
4 bits	1110	07
5 bits	1111 0	08
6 bits	1111 10	09
7 bits	1111 110	0A
8 bits	1111 1110	0B

table ref: https://www.impulseadventure.com/photo/jpeg-huffman-coding.html

JPEG Compression: AC Component

- AC Components are small and normally zero

➢Run Length Encoding (RLE)

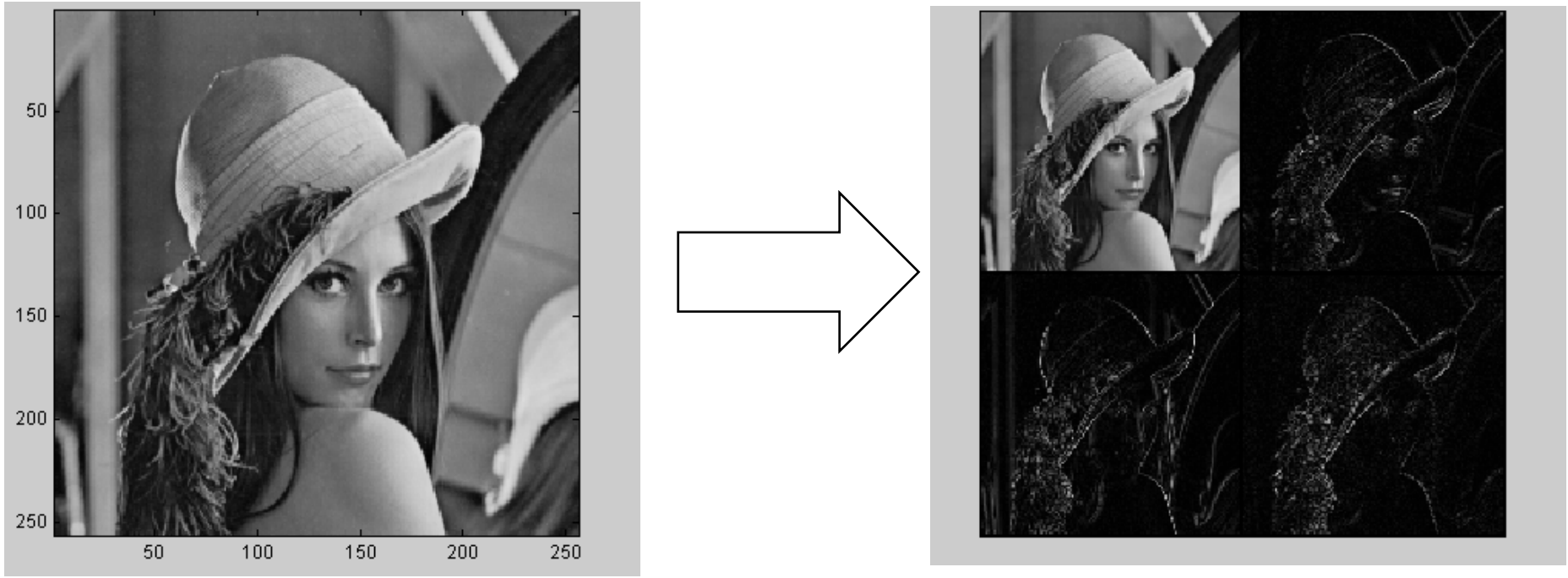
  - 0000020000100002100000 => (5,2)(4,1)(4,2)(0,1)(0,0)

JPEG Problem

- Compression Granularity is in Unit of 8*8

JPEG 2000

- Wavelet Transform

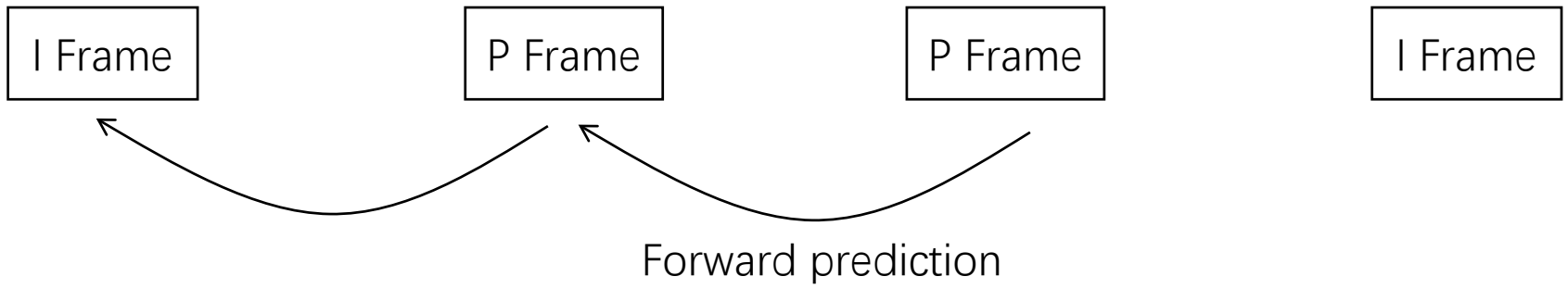


MPEG – Video Compression

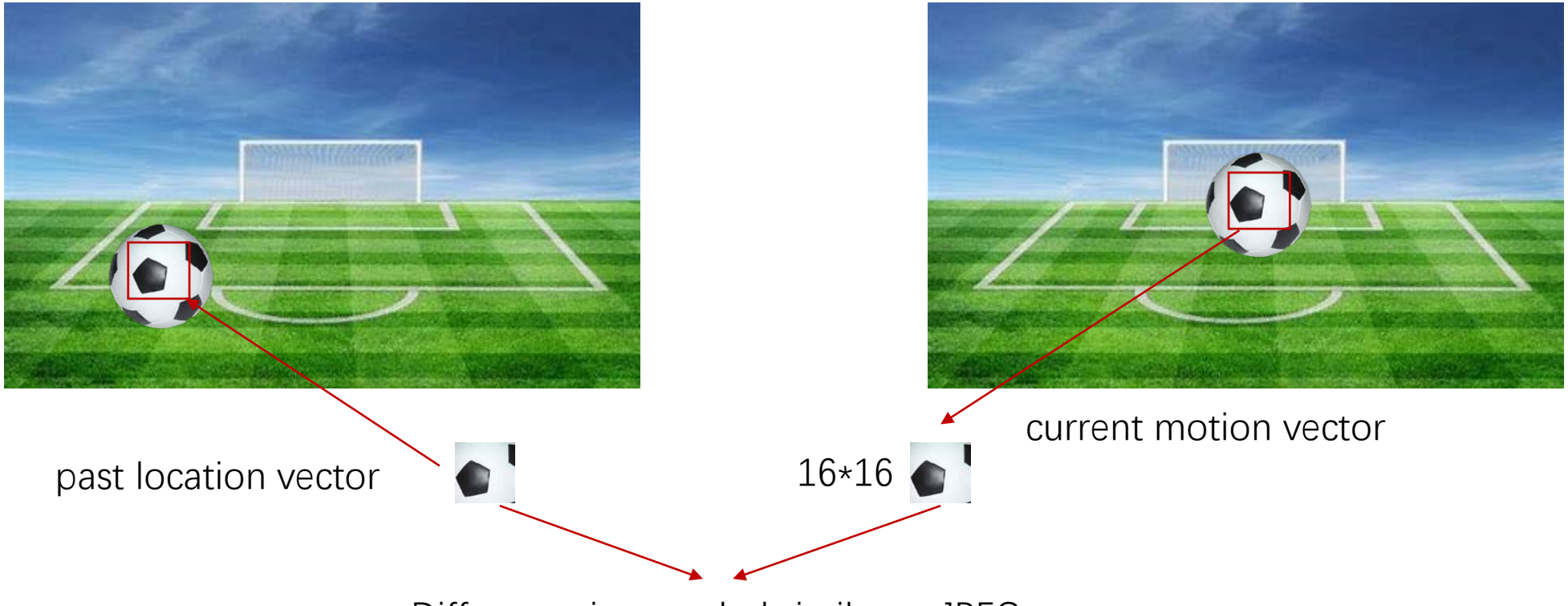
- Filename Extension: MPEG-4 **`.mp4`**
- Moving Pictures Experts Group
- Intuition
  - Adjacent frames are similar and changes are due to foreground motion

MPEG Compression: I Frame and P Frame

- I (intra) Frame
  - Independent frames
  - Coded without reference to other frames (JPEC Compressed)
- P (predictive) Frame
  - Not Independent frames
  - Predicted from a past frame (I or P)

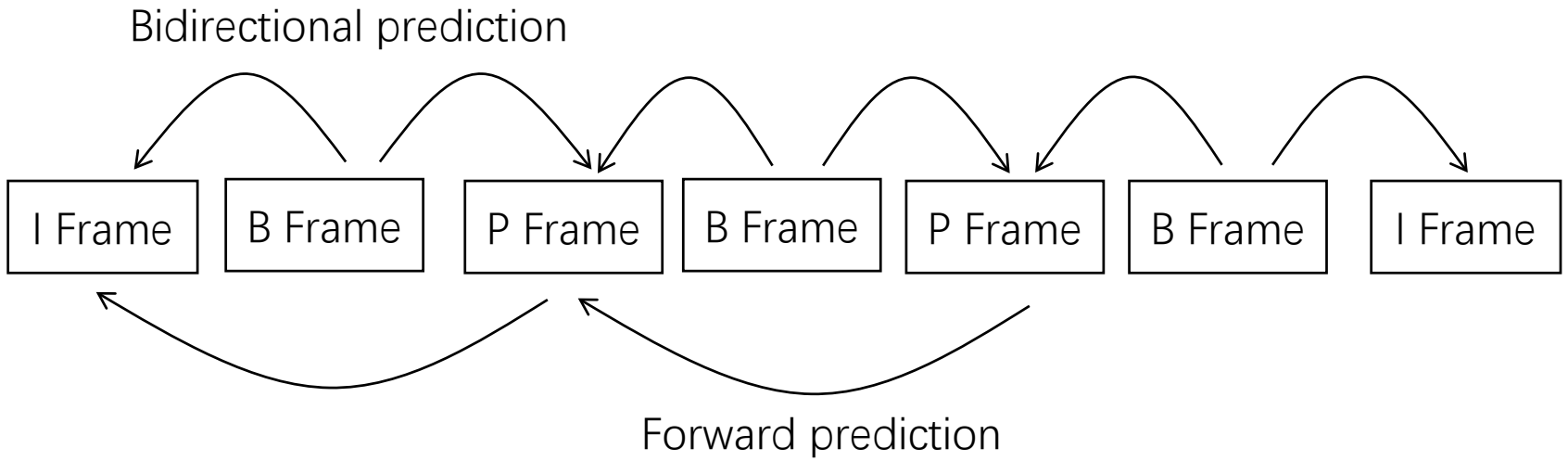


MPEG Compression: Forward Prediction



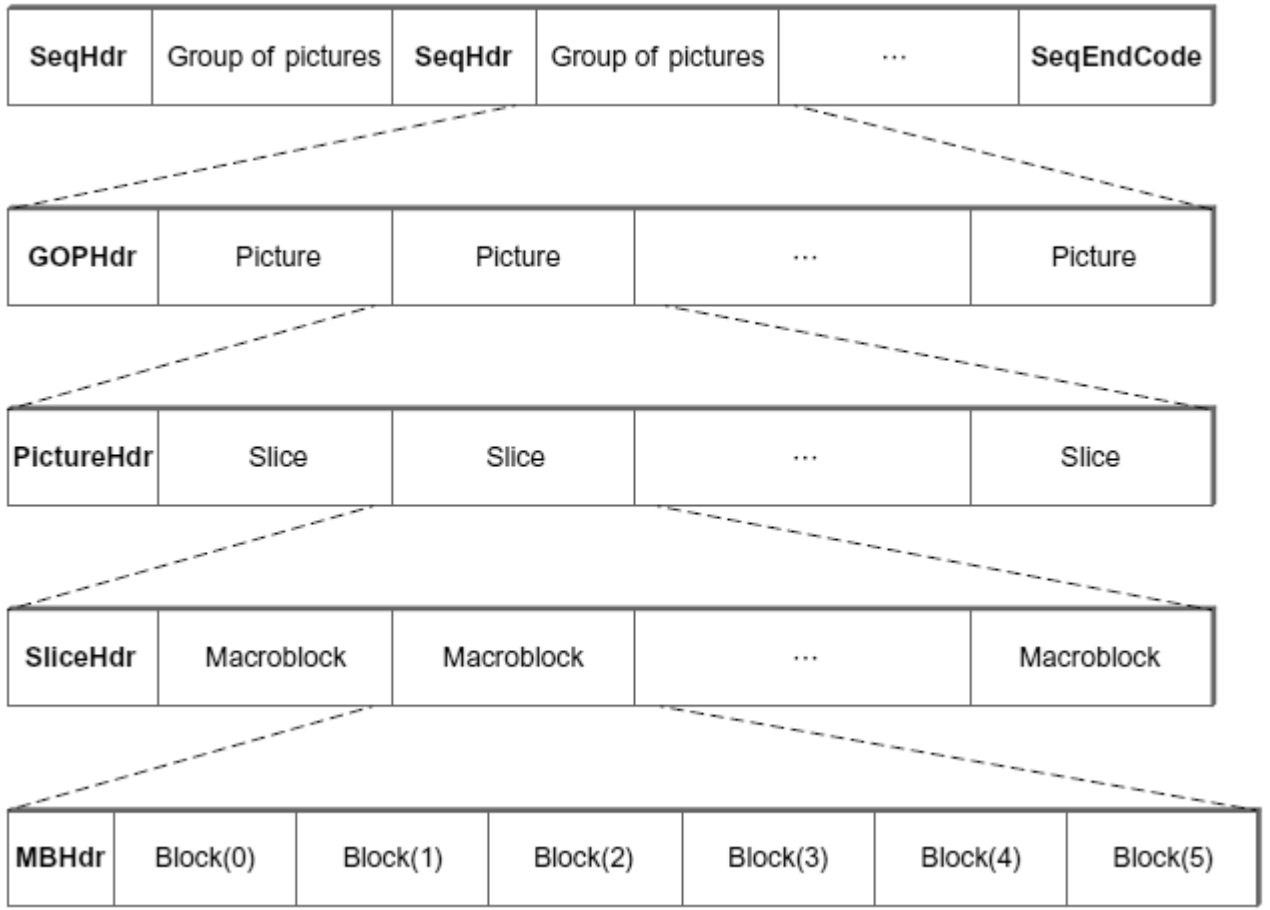
MPEG Compression: B Frame

- B (Bidirectional) Frame
  - Not independent frames
  - Reason: enhance forward prediction
    - The forward I frame might not contain similar information as the B frame
  - Coded with reference to both previous and future frames (I or P)



MPEG over a Network

- A Video Stream



MPEG over a Network

- Frame Sequence
  - Target Seq: IBBBBPBBBBI
  - Transmitting Seq: IPBBBBIBBBB
    - Large Delay
  - For Interactive Videos
    - Only use I and P frames or pure I frames

MP3 – Audio Compression

- Filename Extension: `.mp3`
- A part of MPEG
  - MP3 is introduced in MPEG-1 to encode audio
- Intuition
  - Human ear are less sensitive to high frequency sound
  - Divide audio signal into subbands
  - Compressing subband by allocating different numbers of bits

Reference

- Textbook 7.2