Image Processing

JPEG

정 재 헌

Department of Computer Science and Engineering

Chungnam National University, Korea

실습 소개

- **과목 홈페이지**
  - 충남대학교 사이버 캠퍼스 (http://e-learn.cnu.ac.kr)

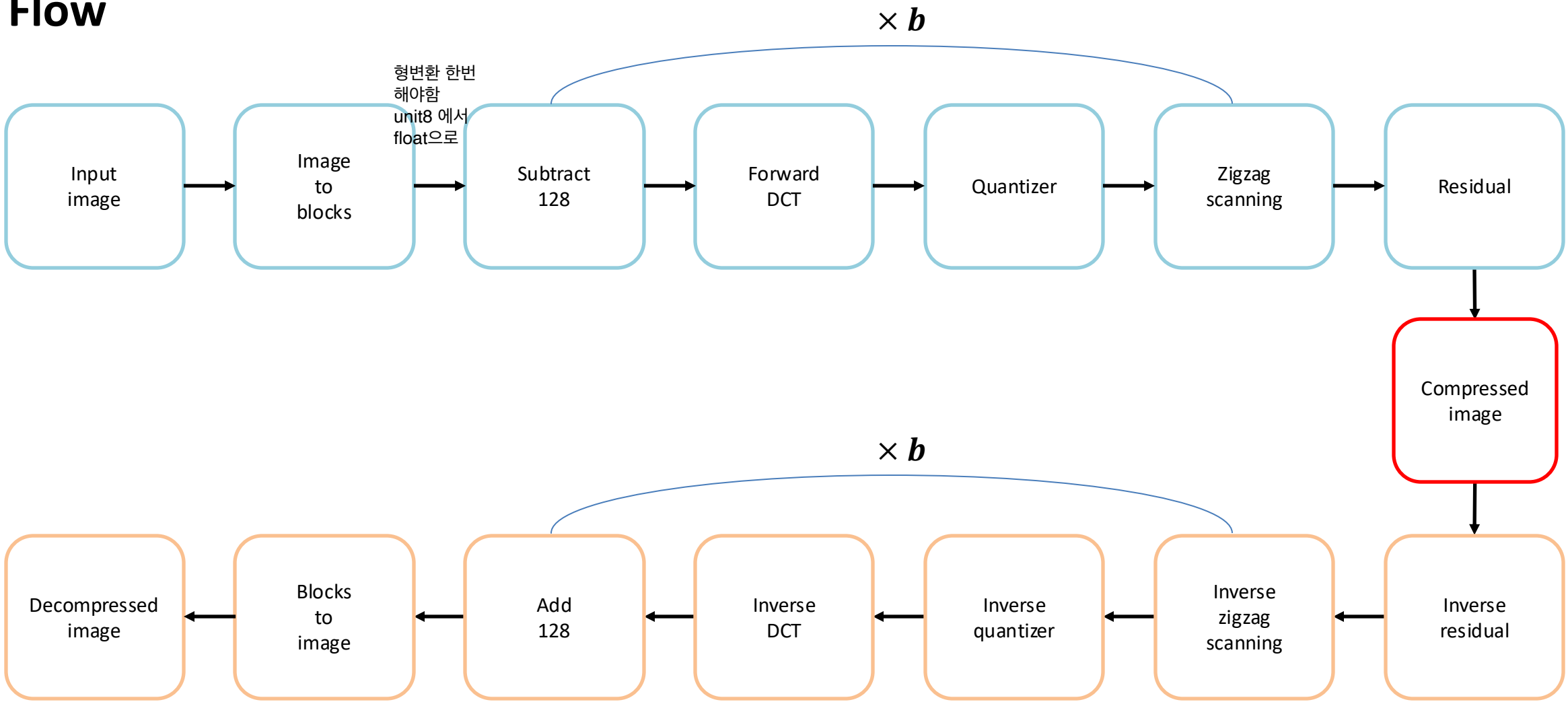
- **TA 연락처**
  - 공대 5호관 531호 컴퓨터비전 연구실
  - 과제 질문은 [IP]를 제목에 붙여 메일로 주세요.
  - 00반
    - 정재헌
    - jaeheonjeong@icloud.com
  - 01반
    - 박광욱
    - ccocco0609@naver.com

목차

- **과제**
  - The JPEG Algorithm
    - Encoding
    - Decoding

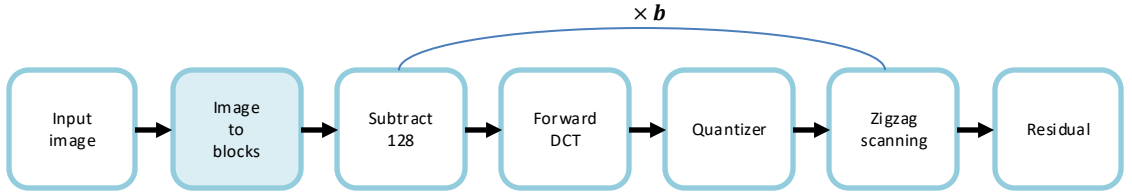
The JPEG Algorithm

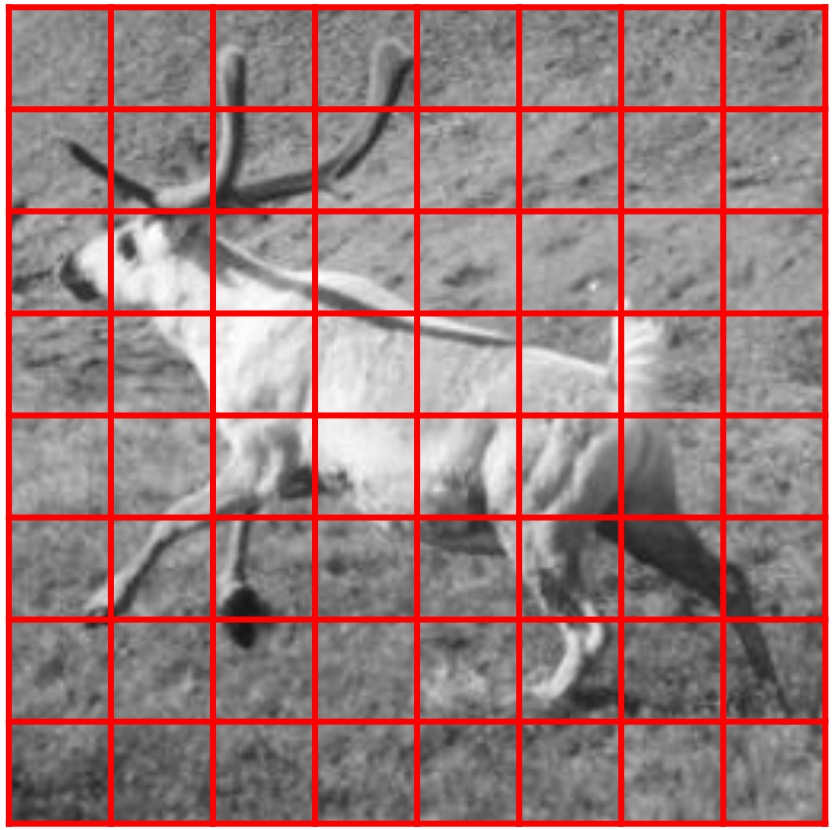
- **Flow**



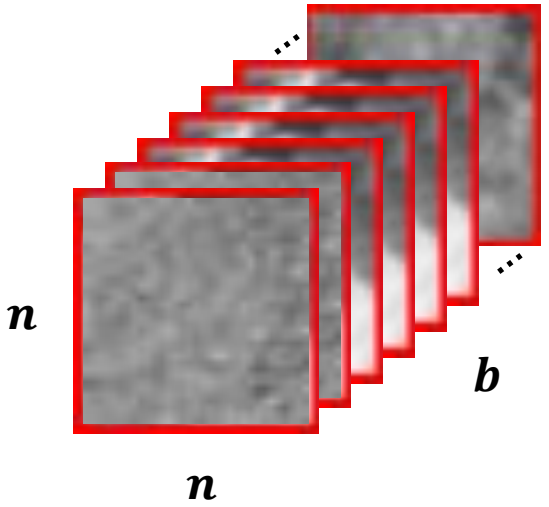
The JPEG Algorithm

- **Encoding**
  - Image to blocks



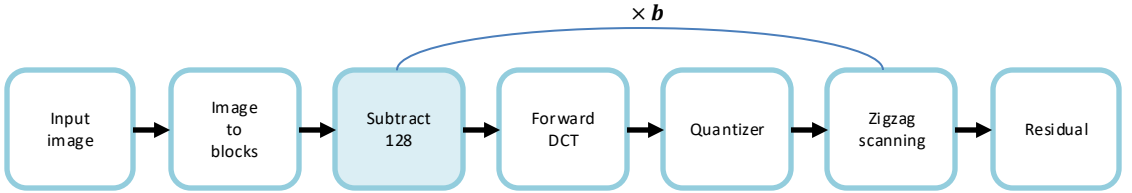


Input image



Blocks

The JPEG Algorithm



- **Encoding**
  - Subtract 128

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

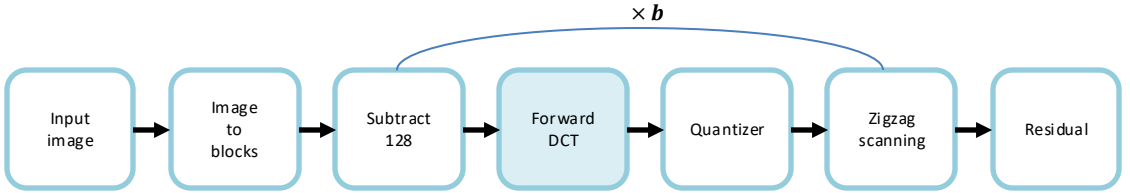
Block

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-60	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

Subtract result

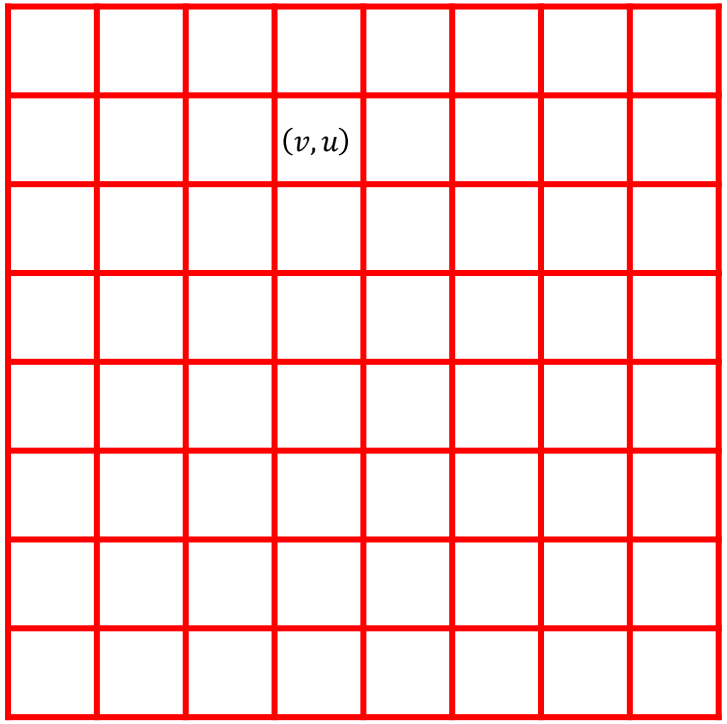
The JPEG Algorithm

- **Encoding**
  - Forward DCT

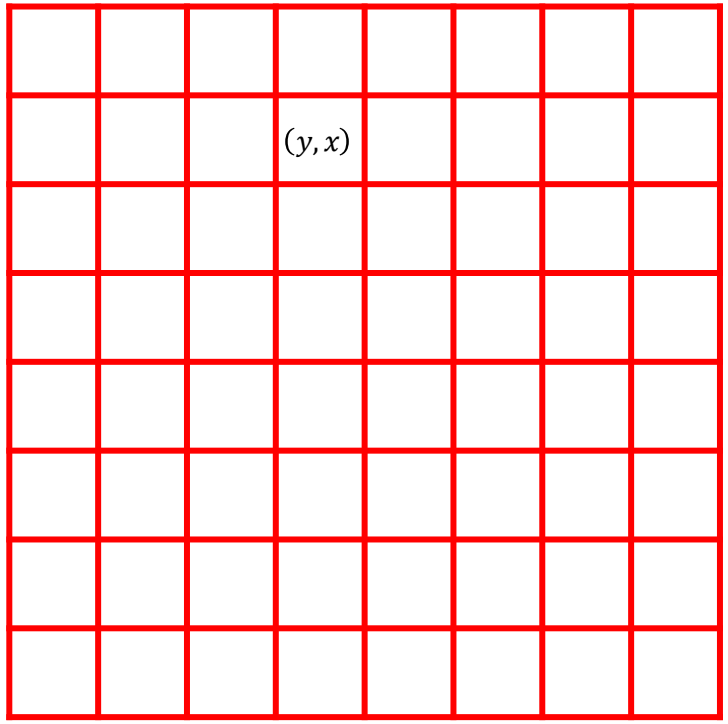


$$F(v,u) = C(v)C(u)\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos\left(\frac{(2y+1)v\pi}{2n}\right)\cos\left(\frac{(2x+1)u\pi}{2n}\right)$$

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

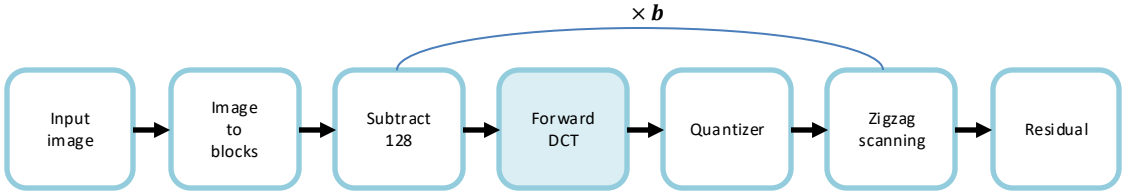


DCT result



subtract result

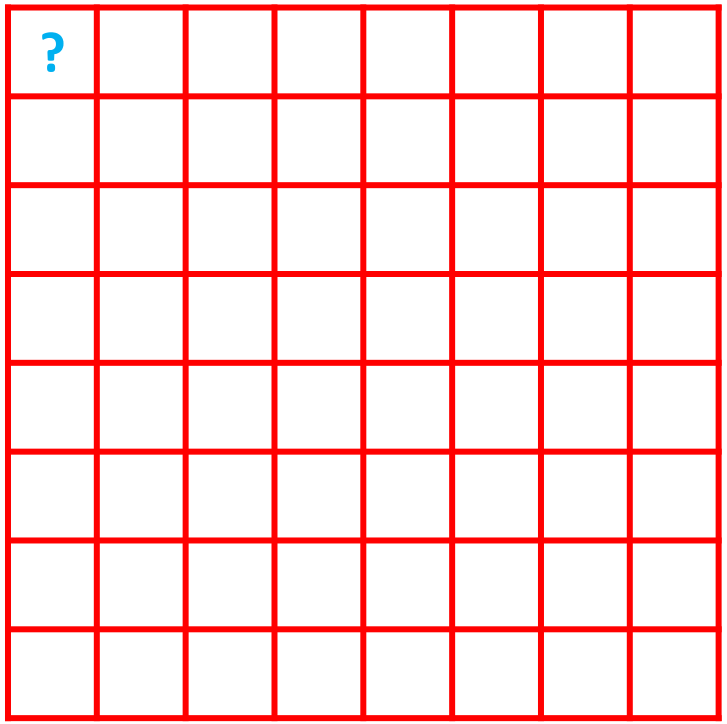
The JPEG Algorithm



- **Encoding**
  - Forward DCT

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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

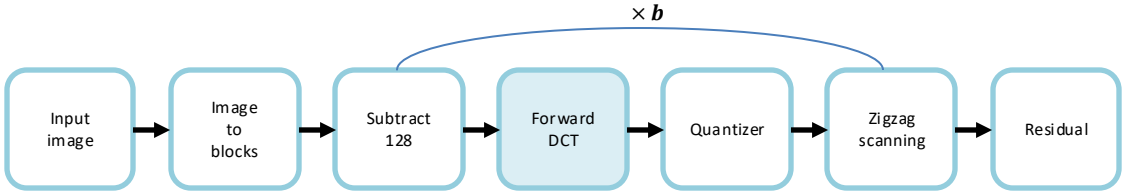


DCT result

$$F(0,0) = C(0)C(0)\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos\left(\frac{(2y+1)0\pi}{2\times 8}\right)\cos\left(\frac{(2x+1)0\pi}{2\times 8}\right)$$

$$F(0,0) = \sqrt{\frac{1}{8}}\sqrt{\frac{1}{8}}\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos(0)\cos(0)$$

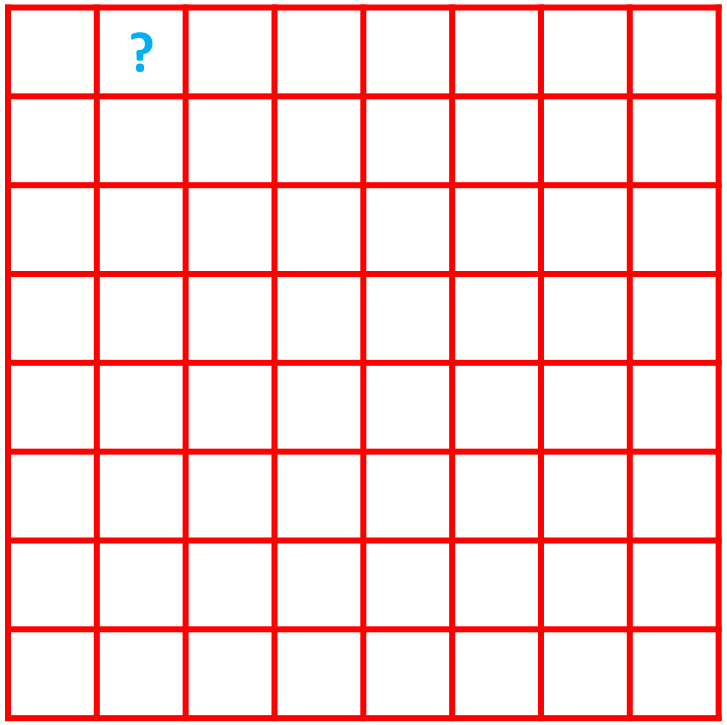
The JPEG Algorithm



- **Encoding**
  - Forward DCT

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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

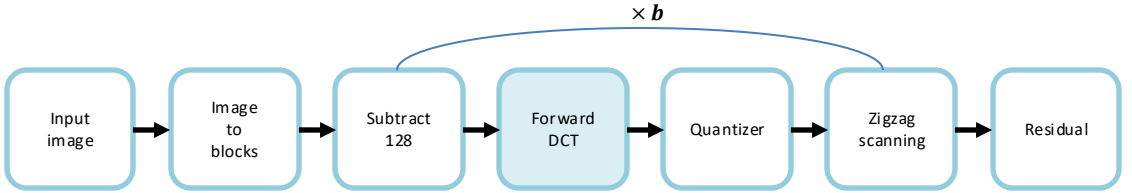


DCT result

$$F(0,1) = C(0)C(1)\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos\left(\frac{(2y+1)0\pi}{2\times 8}\right)\cos\left(\frac{(2x+1)1\pi}{2\times 8}\right)$$

$$F(0,1) = \sqrt{\frac{1}{8}}\sqrt{\frac{2}{8}}\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos(0)\cos\left(\frac{(2x+1)\pi}{16}\right)$$

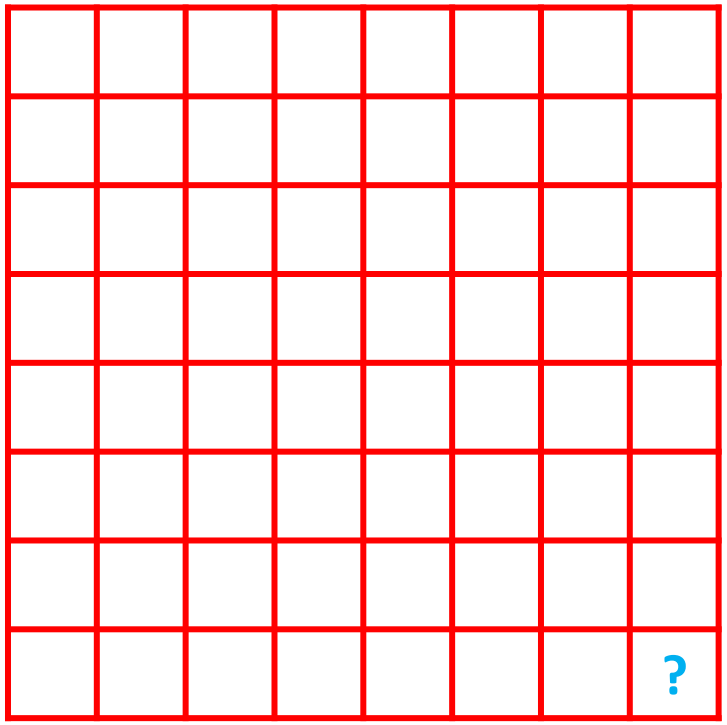
The JPEG Algorithm



- **Encoding**
  - Forward DCT

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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

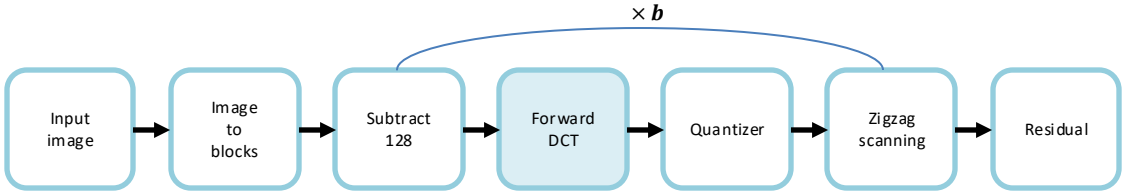


DCT result

$$F(7,7) = C(7)C(7)\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos\left(\frac{(2y+1)7\pi}{2\times 8}\right)\cos\left(\frac{(2x+1)7\pi}{2\times 8}\right)$$

$$F(7,7) = \sqrt{\frac{2}{8}}\sqrt{\frac{2}{8}}\sum_{y=0}^{7}\sum_{x=0}^{7} f(y,x)\cos\left(\frac{(2x+1)7\pi}{16}\right)\cos\left(\frac{(2x+1)7\pi}{16}\right)$$

The JPEG Algorithm



- **Encoding**
  - Forward DCT

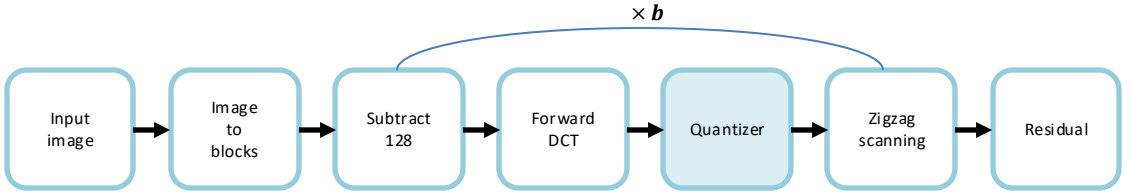
-76.	-73.	-67.	-62.	-58.	-67.	-64.	-55.
-65.	-69.	-62.	-38.	-19.	-43.	-59.	-56.
-66.	-69.	-60.	-15.	16.	-24.	-62.	-55.
-65.	-70.	-57.	-6.	26.	-22.	-58.	-59.
-61.	-67.	-60.	-24.	-2.	-40.	-60.	-58.
-49.	-63.	-68.	-58.	-51.	-60.	-70.	-53.
-43.	-57.	-64.	-69.	-73.	-67.	-63.	-45.
-41.	-49.	-59.	-60.	-63.	-52.	-50.	-34.

Subtract result

-414.	-29.	-61.	25.	54.	-19.	-0.	2.
6.	-20.	-61.	8.	11.	-6.	-6.	6.
-46.	7.	76.	-25.	-29.	10.	6.	-4.
-48.	11.	34.	-14.	-9.	6.	1.	1.
10.	-7.	-12.	-2.	-0.	1.	-4.	1.
-9.	1.	3.	-3.	-0.	0.	1.	0.
-2.	-1.	1.	0.	0.	-3.	1.	-2.
-1.	-0.	-0.	-2.	-0.	-0.	-0.	0.

DCT result

The JPEG Algorithm



- **Encoding**
  - Quantizer

round

-414.	-29.	-61.	25.	54.	-19.	-0.	2.
6.	-20.	-61.	8.	11.	-6.	-6.	6.
-46.	7.	76.	-25.	-29.	10.	6.	-4.
-48.	11.	34.	-14.	-9.	6.	1.	1.
10.	-7.	-12.	-2.	-0.	1.	-4.	1.
-9.	1.	3.	-3.	-0.	0.	1.	0.
-2.	-1.	1.	0.	0.	-3.	1.	-2.
-1.	-0.	-0.	-2.	-0.	-0.	-0.	0.

Subtract result

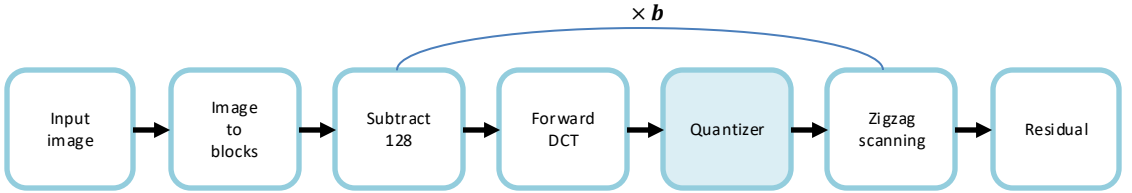
/

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	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization matrix

The JPEG Algorithm

- **Encoding**
  - Quantizer



-414.	-29.	-61.	25.	54.	-19.	-0.	2.
6.	-20.	-61.	8.	11.	-6.	-6.	6.
-46.	7.	76.	-25.	-29.	10.	6.	-4.
-48.	11.	34.	-14.	-9.	6.	1.	1.
10.	-7.	-12.	-2.	-0.	1.	-4.	1.
-9.	1.	3.	-3.	-0.	0.	1.	0.
-2.	-1.	1.	0.	0.	-3.	1.	-2.
-1.	-0.	-0.	-2.	-0.	-0.	-0.	0.

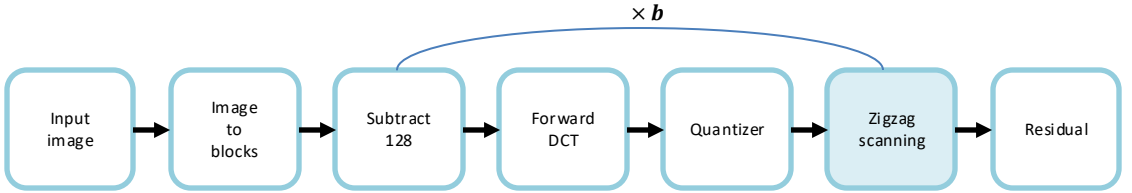
Subtract result

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization result

The JPEG Algorithm

- **Encoding**
  - Zigzag scanning



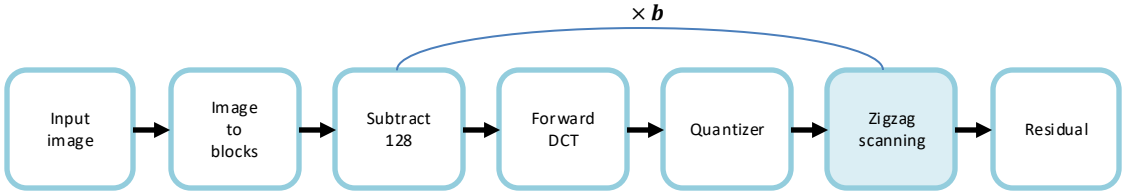
-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization result

[-26, -3, 1, -3, -2, -6, 2, -4,
1, -3, 1, 1, 5, 0, 2, 0,
0, -1, 2, 0, 0, 0, 0, 0,
0, -1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0]

The JPEG Algorithm

- **Encoding**
  - Zigzag scanning



-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization result

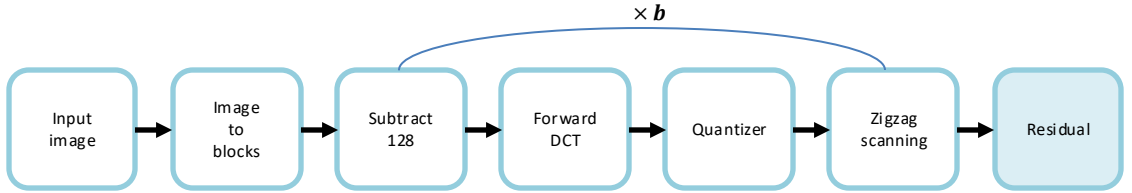
[-26, -3, 1, -3, -2, -6, 2, -4,
1, -3, 1, 1, 5, 0, 2, 0,
0, -1, 2, 0, 0, 0, 0, 0,
0, -1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0]

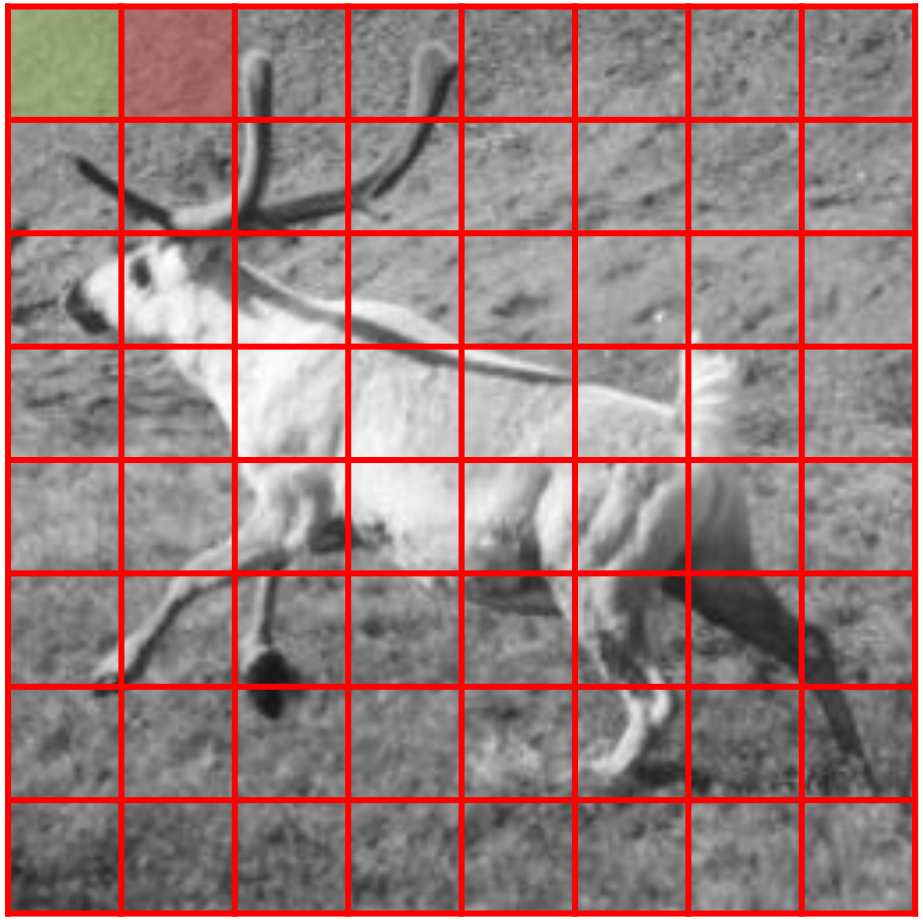
[-26, -3, 1, -3, -2, -6, 2, -4,
1, -3, 1, 1, 5, 0, 2, 0,
0, -1, 2, 0, 0, 0, 0, 0,
0, -1, 'EOB']

← Compressed Block

The JPEG Algorithm

- **Encoding**
  - Residual





Quantization result

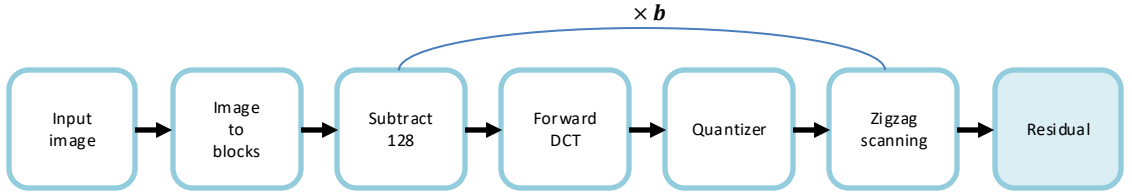
Block0 = [-15, -1, 1, -2, -1, -3, 2, -4, 'EOB']

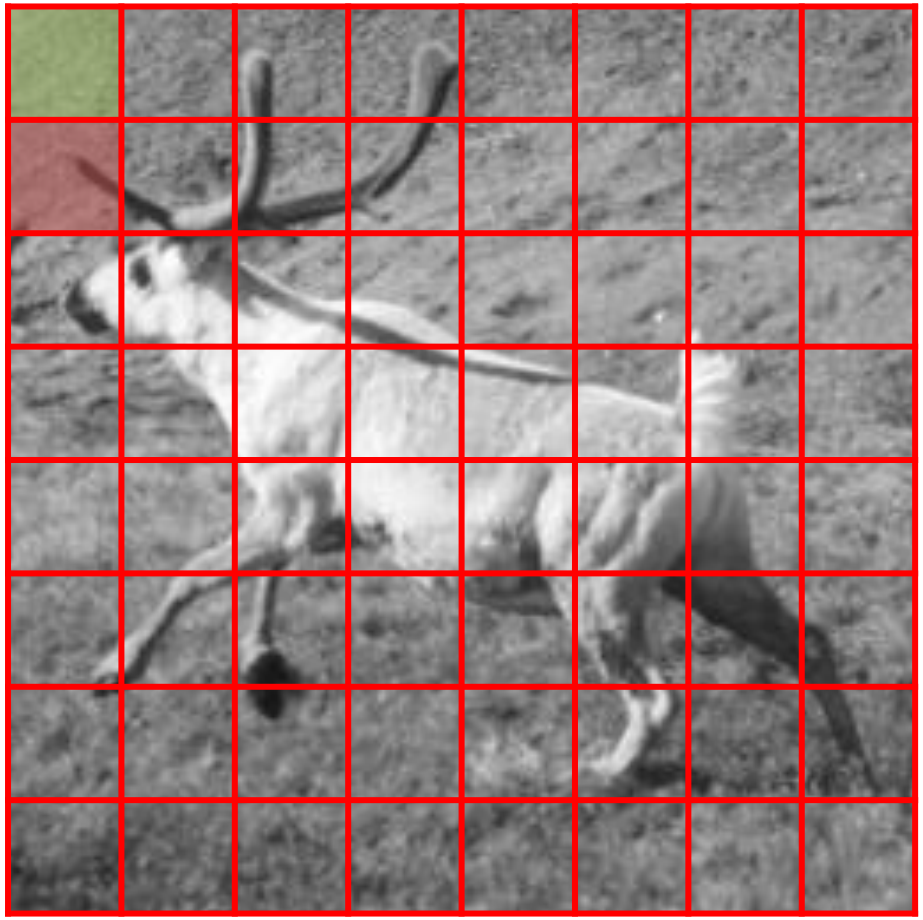
Block1 = [-26, -3, 1, -3, -2, -6, 2, -4, 1, -3, 1, 1, 5, 0, 2, 0, 0, -1, 2, 0, 0, 0, 0, 0, 0, -1, 'EOB']

Block1 = [-8, -2, 0, -1, -1, -3, 0, 0, 1, -3, 1, 1, 5, 0, 2, 0, 0, -1, 2, 0, 0, 0, 0, 0, 0, -1, 'EOB']

The JPEG Algorithm

- **Encoding**
  - Residual





Quantization result

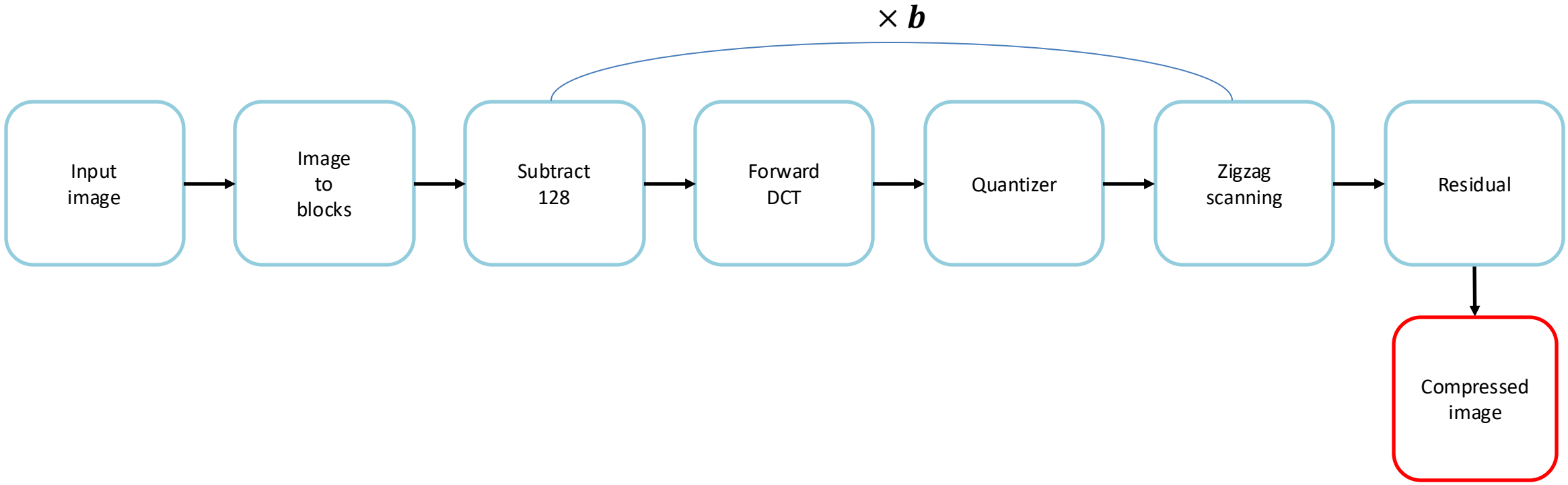
Block0 = [-15, -1, 1, -2, -1, -3, 2, -4, 'EOB']

Block8 = [-15, -1, 1, -2, -2, -6, 2, -4, 1, -3, 1, 1, 5, 0, 2, 0, 1, 'EOB']

Block8 = [0, 0, 0, 0, -1, -3, 0, 0, 1, -3, 1, 1, 5, 0, 2, 0, 1, 'EOB']

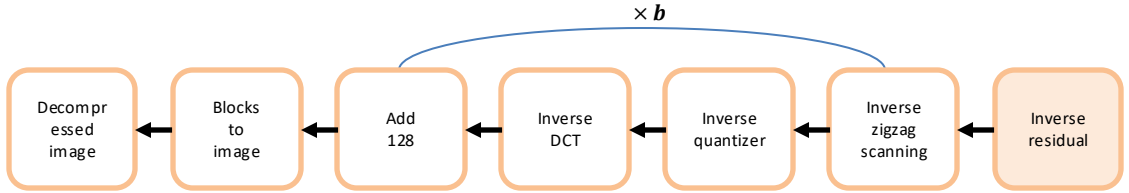
The JPEG Algorithm

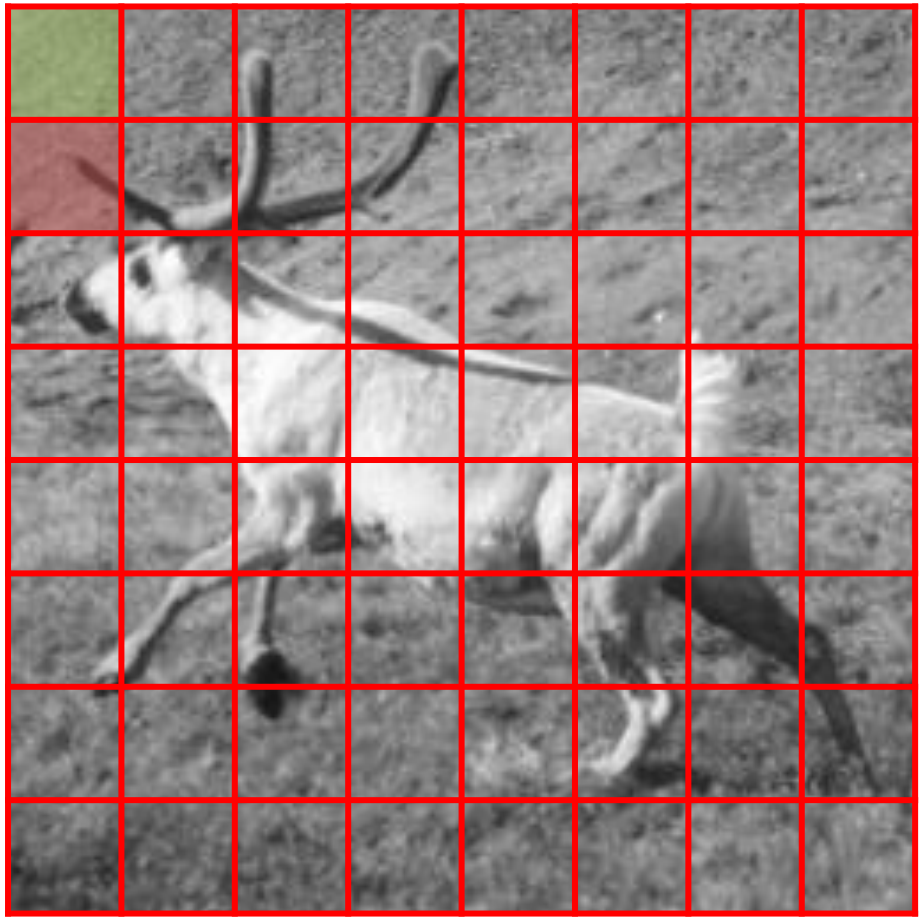
- **Encoding**
  - Compressed image, input image shape를 .npy확장자를 가진 파일로 저장



The JPEG Algorithm

- **Decoding**
  - Inverse residual





Quantization result

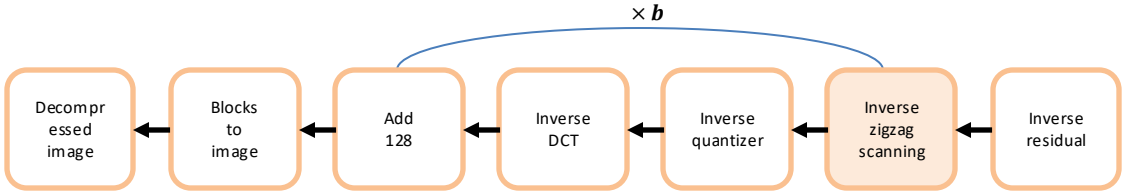
Block0 = [-15, -1, 1, -2, -1, -3, 2, -4, 'EOB']

Block8 = [0, 0, 0, 0, -1, -3, 0, 0, 1, -3, 1, 1, 5, 0, 2, 0, 1, 'EOB']

Block8 = [-15, -1, 1, -2, -2, -6, 2, -4, 1, -3, 1, 1, 5, 0, 2, 0, 1, 'EOB']

The JPEG Algorithm

- **Decoding**
  - Inverse zigzag scanning



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

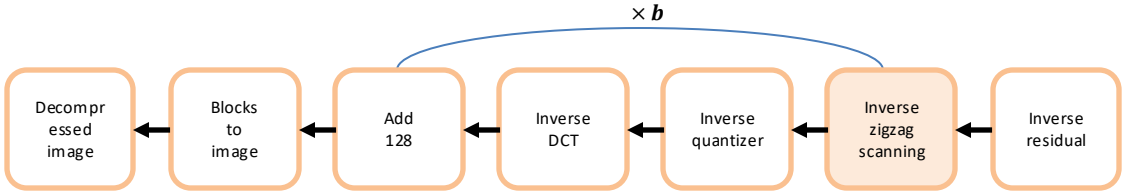
Inverse scanning result

[-26, -3, 1, -3, -2, -6, 2, -4,
1, -3, 1, 1, 5, 0, 2, 0,
0, -1, 2, 0, 0, 0, 0, 0,
0, -1, ‘EOB’]

← Compressed Block

The JPEG Algorithm

- **Decoding**
  - Inverse zigzag scanning



-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

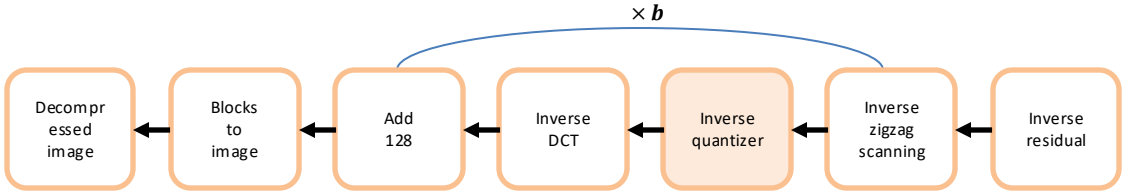
Inverse scanning result

[-26, -3, 1, -3, -2, -6, 2, -4,
1, -3, 1, 1, 5, 0, 2, 0,
0, -1, 2, 0, 0, 0, 0, 0,
0, -1, ‘EOB’]

← Compressed Block

The JPEG Algorithm

- **Decoding**
  - Inverse quantizer



-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Inverse scanning result

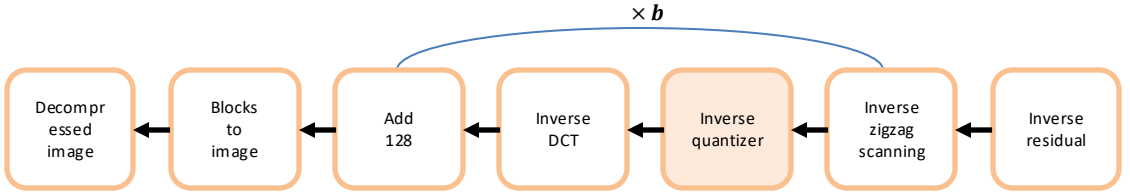
*

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	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization matrix

The JPEG Algorithm

- **Decoding**
  - Inverse quantizer



-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

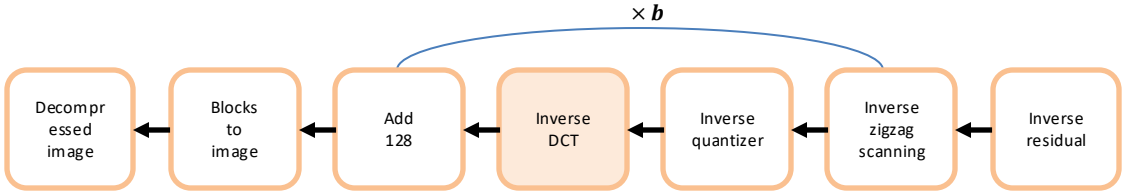
Inverse scanning result

-416	-33	-60	32	48	0	0	0
12	-24	-56	0	0	0	0	0
-42	13	80	-24	-40	0	0	0
-42	17	44	0	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Inverse quantization result

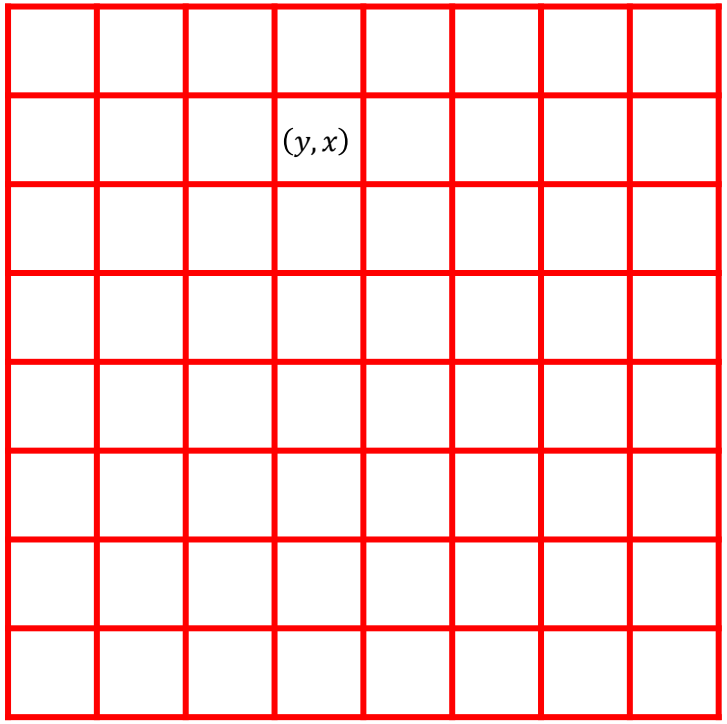
The JPEG Algorithm

- **Decoding**
  - Inverse DCT

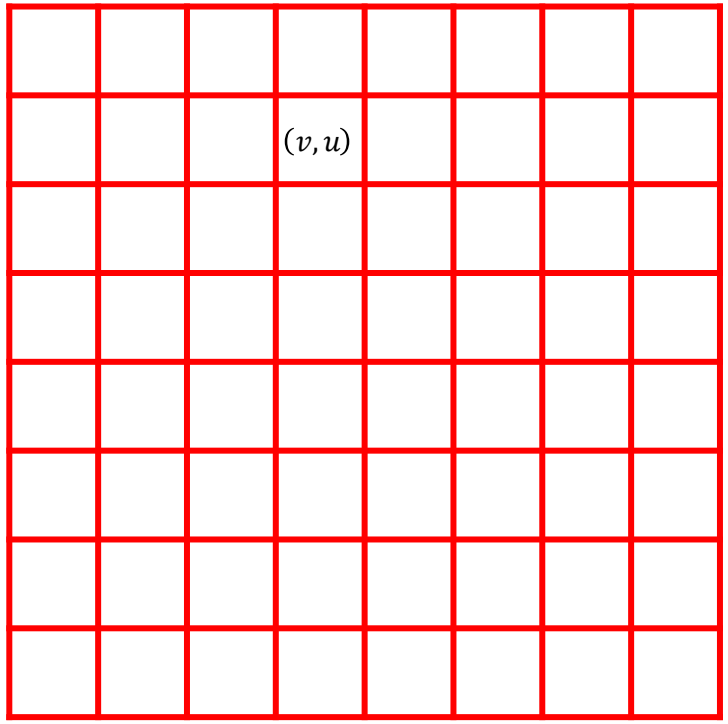


$$f(y,x) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{(2y+1)v\pi}{2n}\right)\cos\left(\frac{(2x+1)u\pi}{2n}\right)$$

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

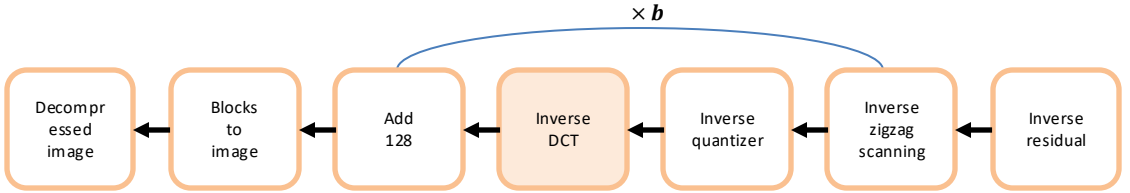


Inverse DCT result



Inverse quantization result

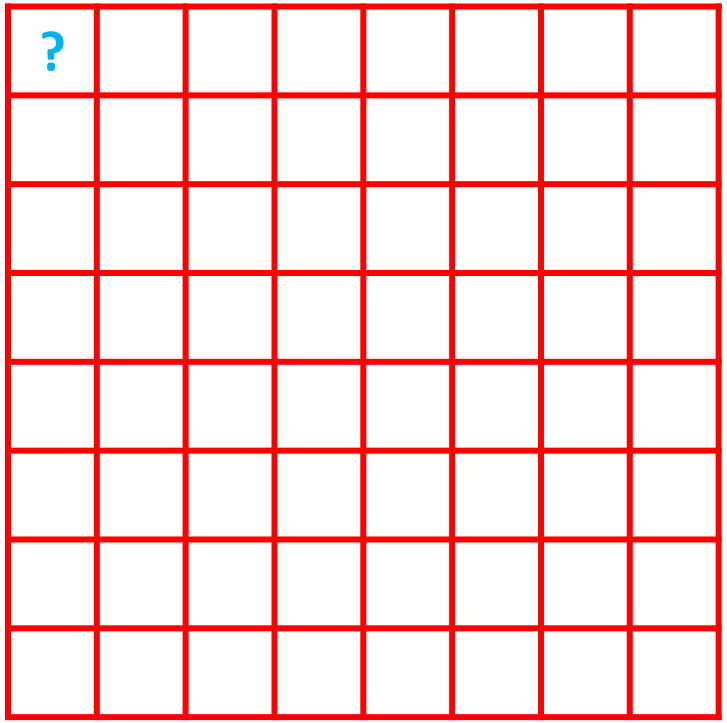
The JPEG Algorithm



- **Decoding**
  - Inverse DCT

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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

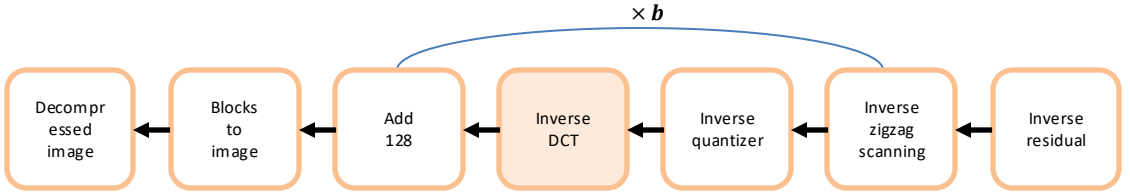


Inverse DCT result

$$f(0,0) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{(2\times 0+1)v\pi}{2\times 8}\right)\cos\left(\frac{(2\times 0+1)u\pi}{2\times 8}\right)$$

$$f(0,0) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{v\pi}{16}\right)\cos\left(\frac{u\pi}{16}\right)$$

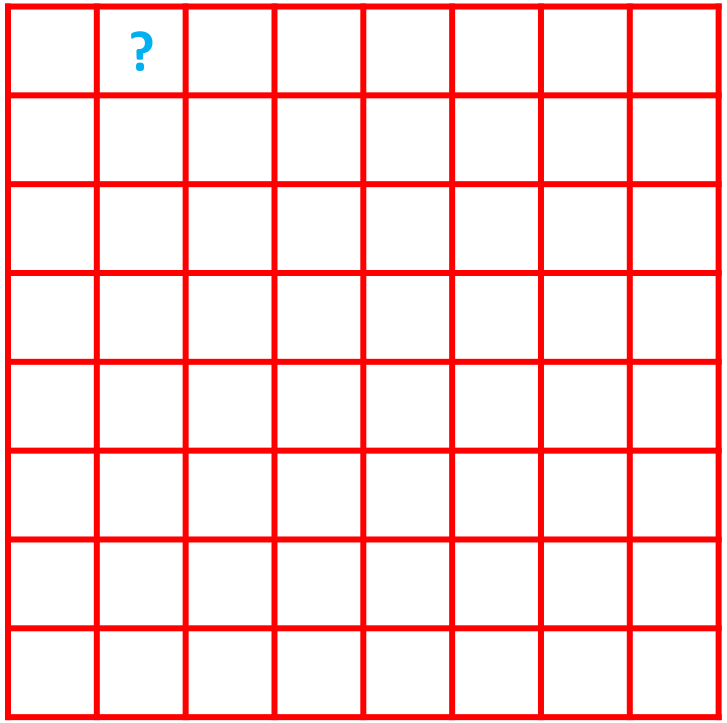
The JPEG Algorithm



- **Decoding**
  - Inverse DCT

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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$



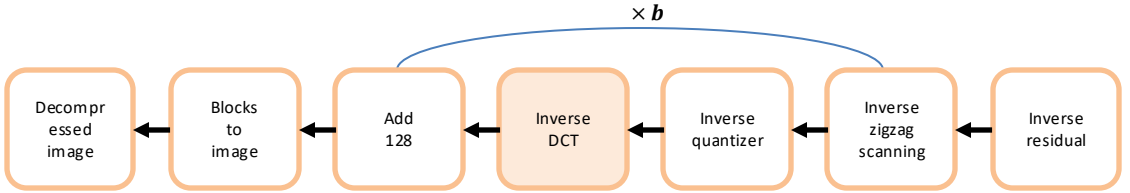
Inverse DCT result

$$f(0,1) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{(2\times 0+1)v\pi}{2\times 8}\right)\cos\left(\frac{(2\times 1+1)u\pi}{2\times 8}\right)$$

$$f(0,1) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{v\pi}{16}\right)\cos\left(\frac{3u\pi}{16}\right)$$

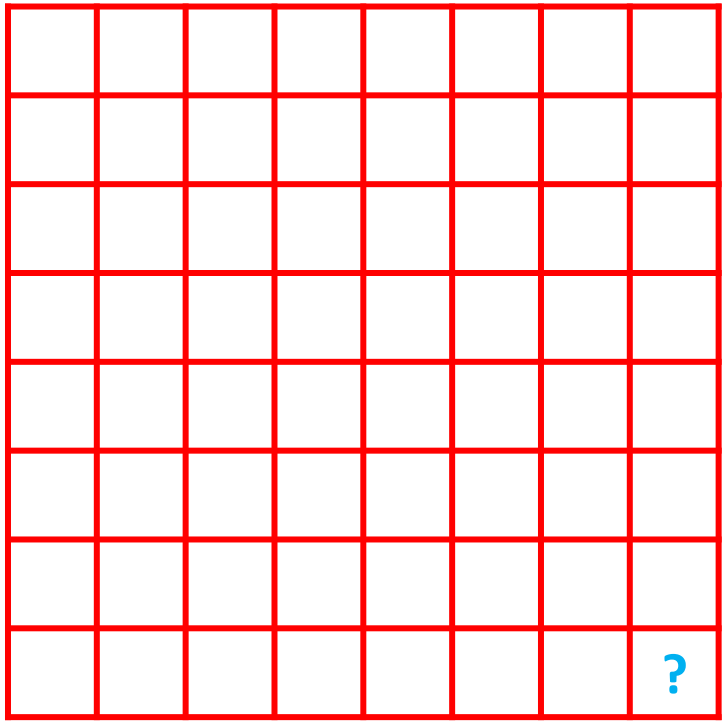
The JPEG Algorithm

- **Decoding**
  - Inverse DCT



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

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$



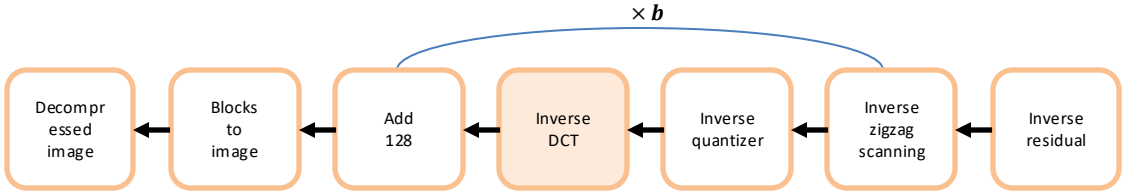
Inverse DCT result

$$f(7,7) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{(2\times 7+1)v\pi}{2\times 8}\right)\cos\left(\frac{(2\times 7+1)u\pi}{2\times 8}\right)$$

$$f(7,7) = \sum_{u=0}^{7}\sum_{v=0}^{7} F(v,u)C(v)C(u)\cos\left(\frac{15v\pi}{16}\right)\cos\left(\frac{15u\pi}{16}\right)$$

The JPEG Algorithm

- **Decoding**
  - Inverse DCT



-416	-33	-60	32	48	0	0	0
12	-24	-56	0	0	0	0	0
-42	13	80	-24	-40	0	0	0
-42	17	44	0	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

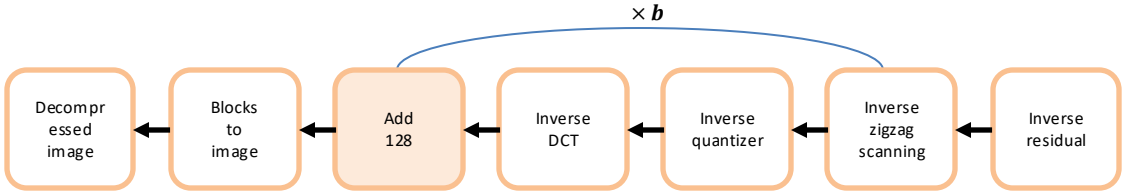
Inverse quantization result

-63	-63	-64	-65	-63	-58	-55	-53
-73	-73	-60	-39	-31	-42	-54	-59
-76	-79	-53	-7	7	-22	-52	-61
-64	-78	-54	1	18	-19	-53	-58
-49	-74	-66	-23	-9	-38	-61	-58
-44	-70	-76	-56	-47	-61	-67	-58
-43	-59	-70	-69	-65	-65	-60	-51
-42	-48	-57	-65	-64	-56	-47	-41

Inverse DCT result

The JPEG Algorithm

- **Decoding**
  - Add 128



-63	-63	-64	-65	-63	-58	-55	-53
-73	-73	-60	-39	-31	-42	-54	-59
-76	-79	-53	-7	7	-22	-52	-61
-64	-78	-54	1	18	-19	-53	-58
-49	-74	-66	-23	-9	-38	-61	-58
-44	-70	-76	-56	-47	-61	-67	-58
-43	-59	-70	-69	-65	-65	-60	-51
-42	-48	-57	-65	-64	-56	-47	-41

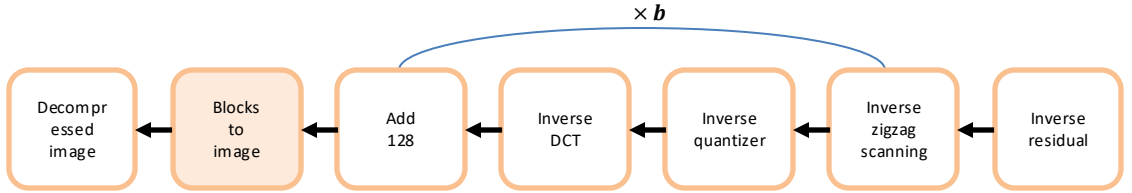
Inverse DCT result

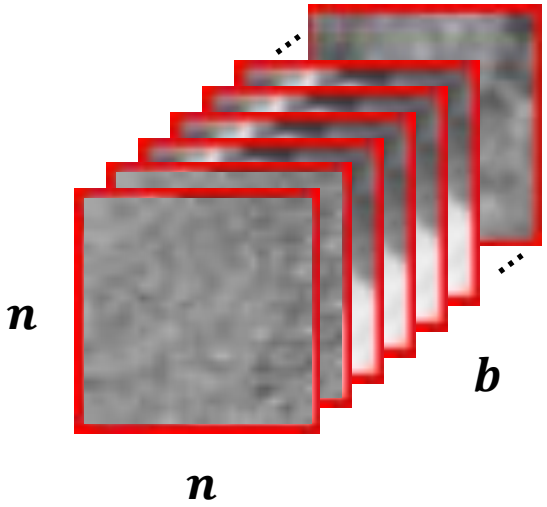
65	65	64	63	65	70	73	75
55	55	68	89	97	86	74	69
52	49	75	121	135	106	76	67
64	50	74	129	146	109	75	70
79	54	62	105	119	90	67	70
84	58	52	72	81	67	61	70
85	69	58	59	63	63	68	77
86	80	71	63	64	72	81	87

Decompressed block

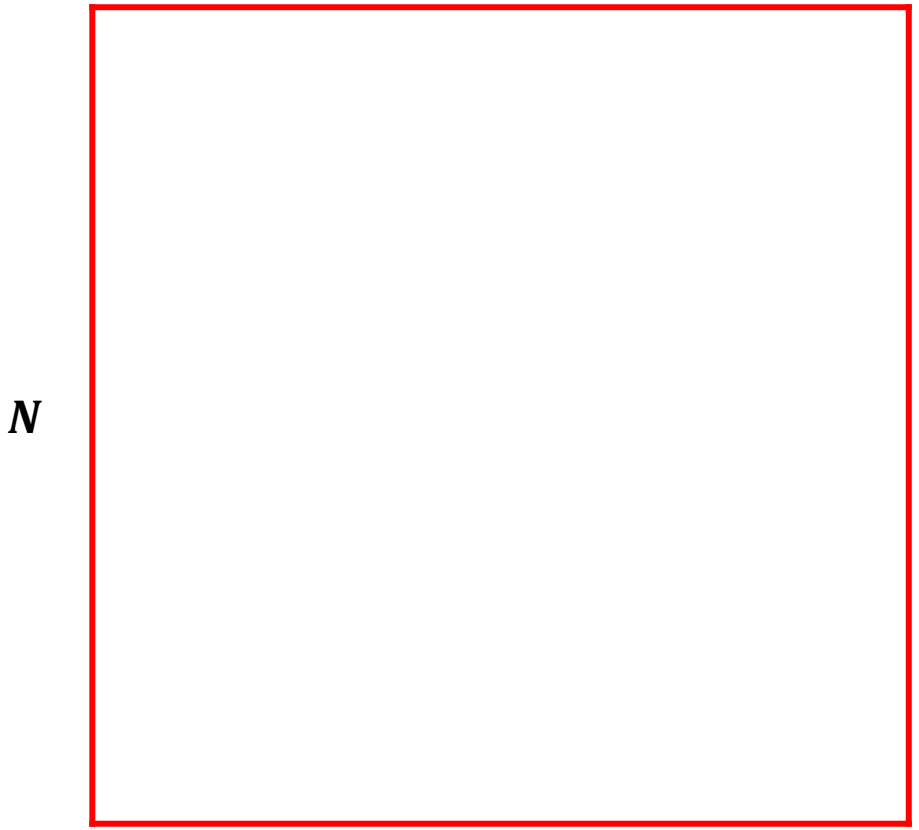
The JPEG Algorithm

- **Decoding**
  - Blocks to image





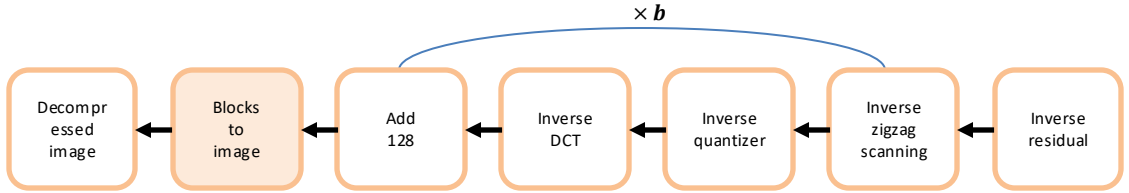
Decompressed blocks

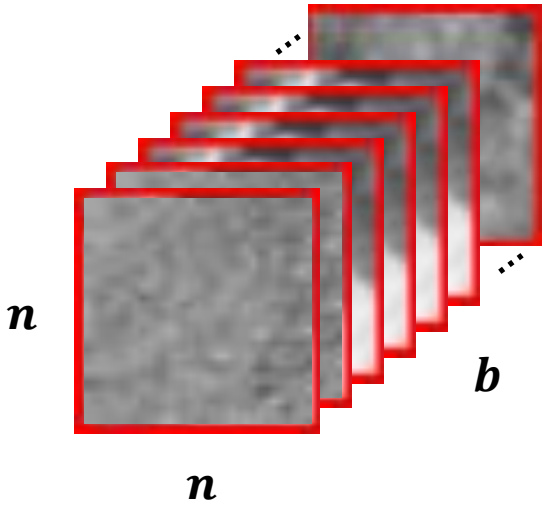


Decompressed image
(빈 배열)

The JPEG Algorithm

- **Decoding**
  - Blocks to image





Decompressed blocks

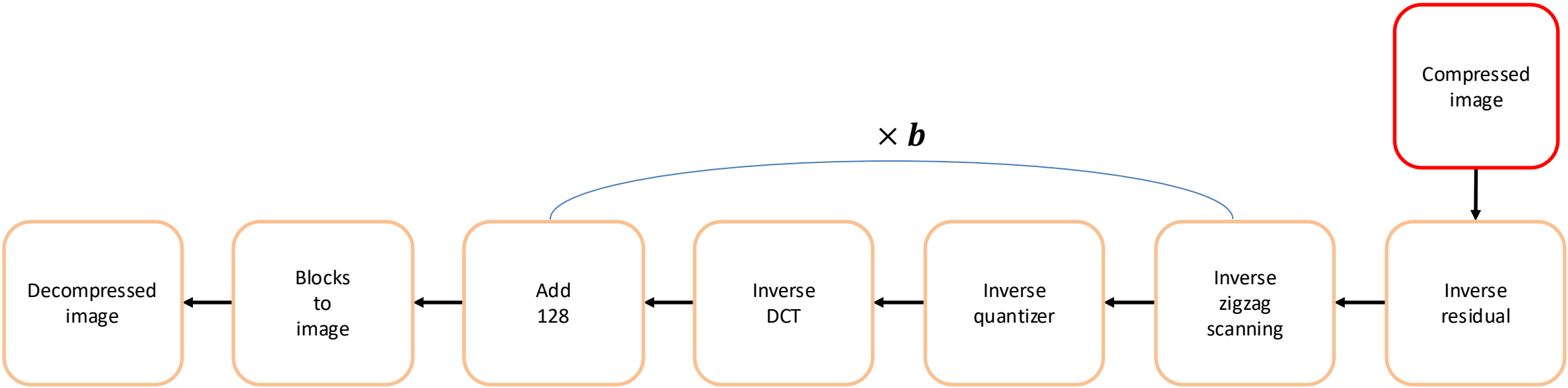
N

N

Decompressed image

The JPEG Algorithm

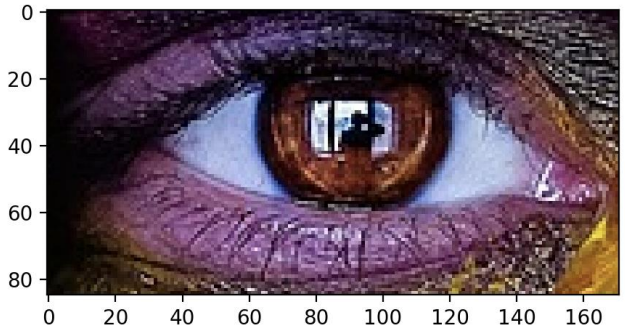
- **Decoding**

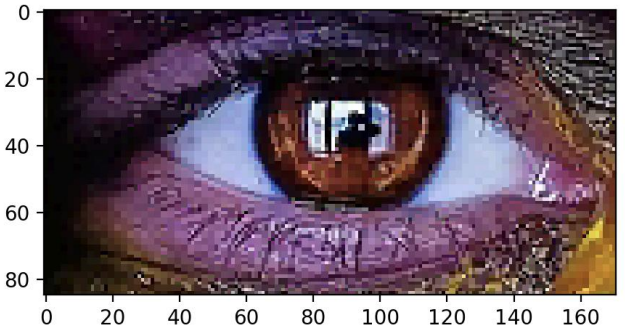


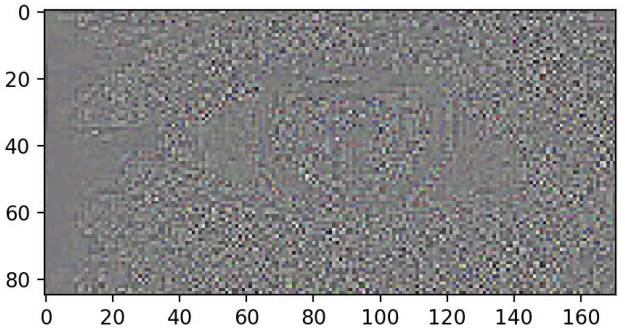
The JPEG Algorithm

- **Quantization matrix의 scale에 따른 결과 (block size : 2 x 2)**

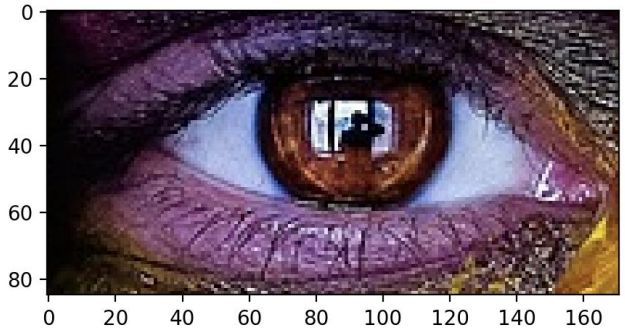
Scale : 1

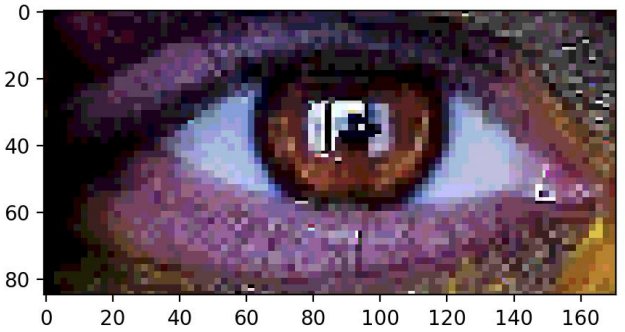


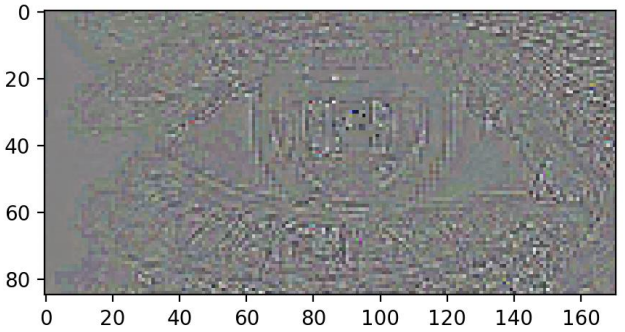




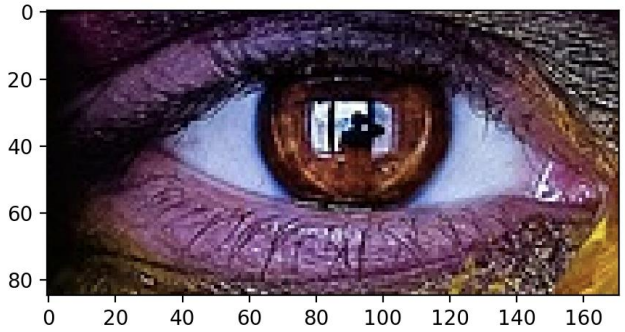
Scale : 5

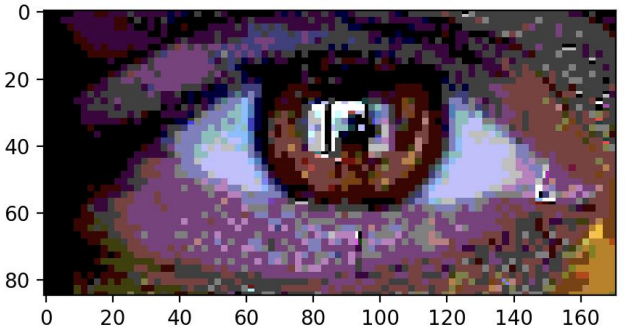


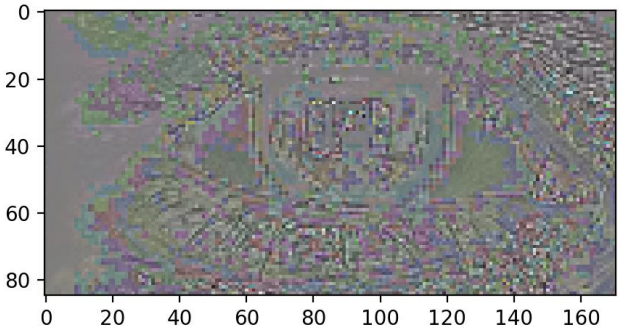




Scale : 10



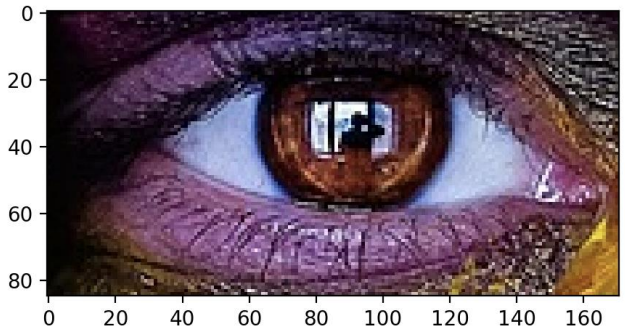
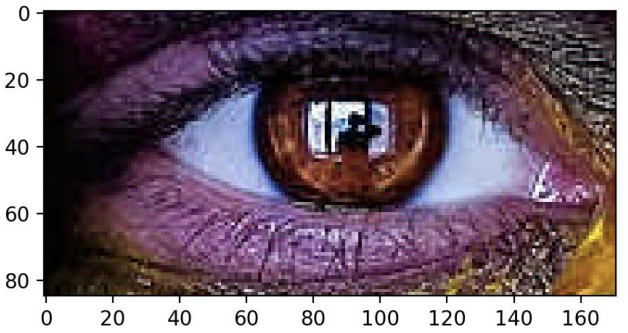
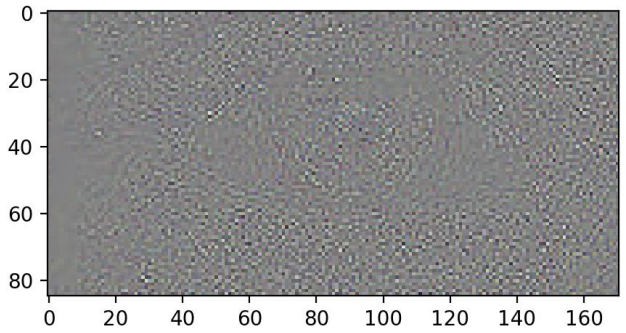




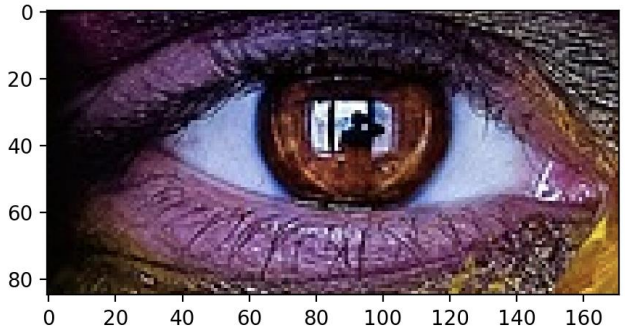
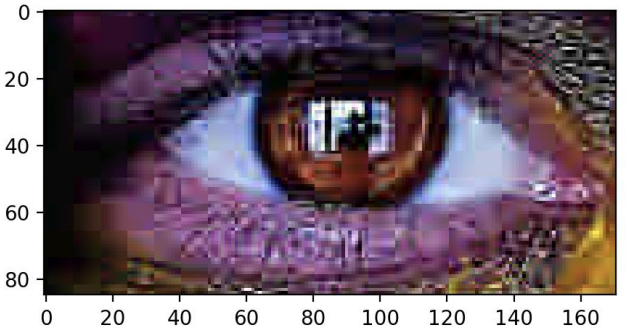
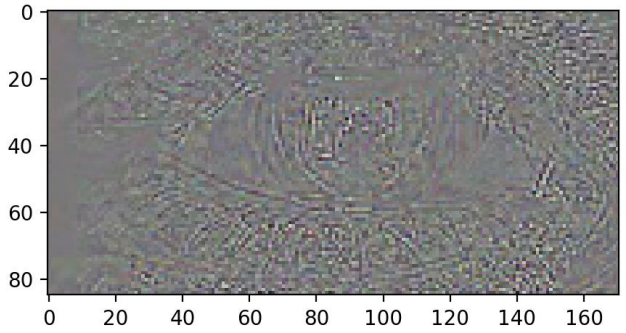
The JPEG Algorithm

- **Quantization matrix의 scale에 따른 결과 (block size : 8 x 8)**

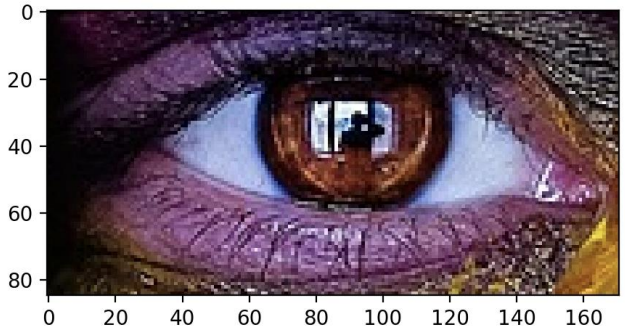
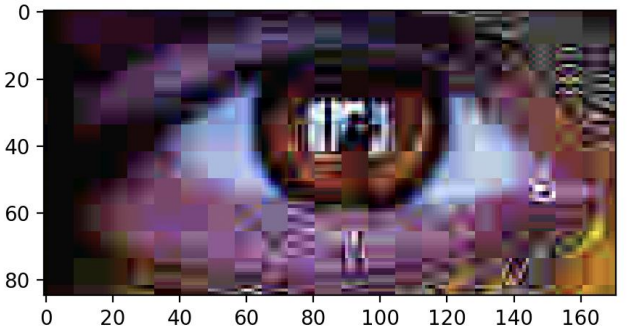
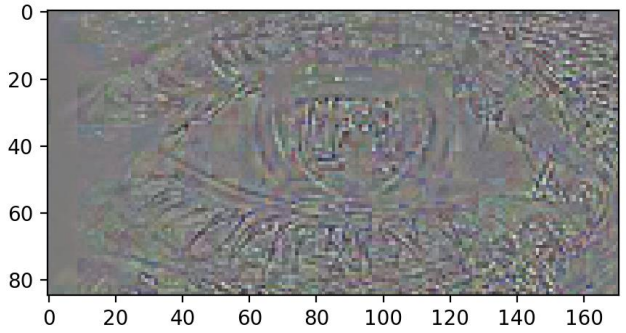
Scale : 1

Scale : 5

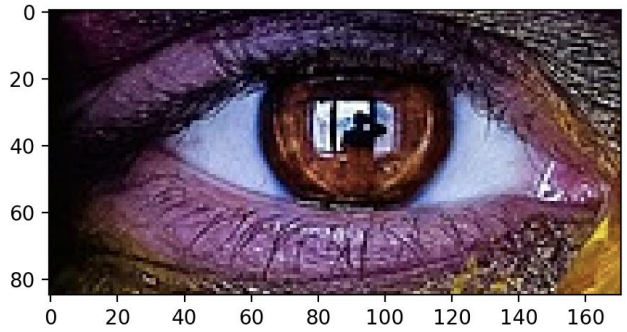
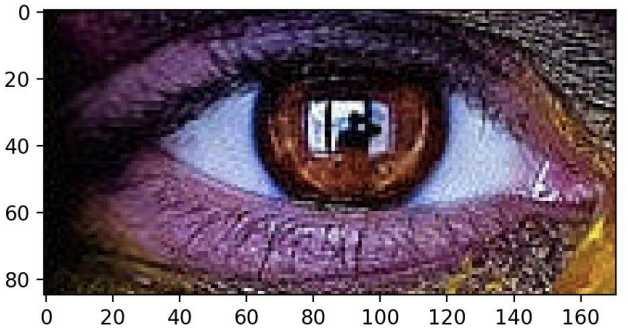
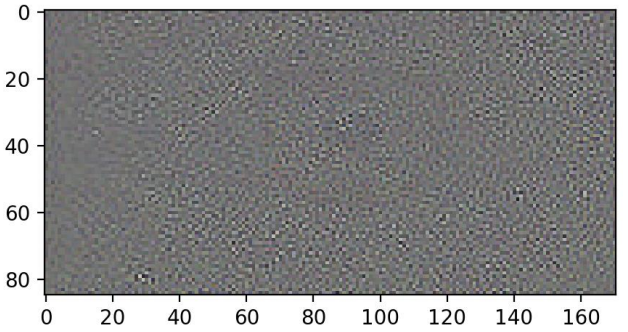
Scale : 10

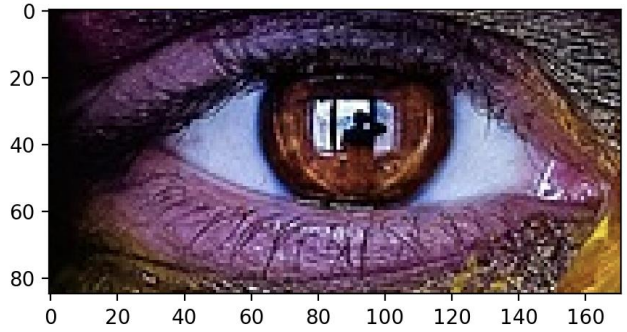
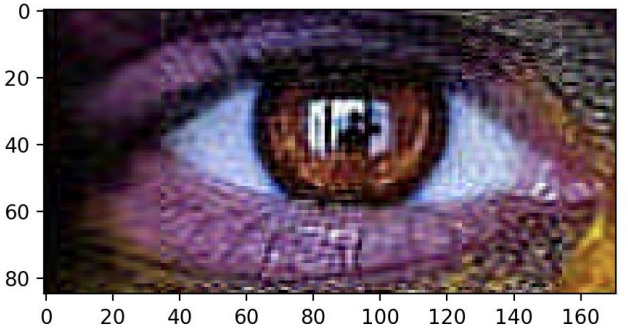
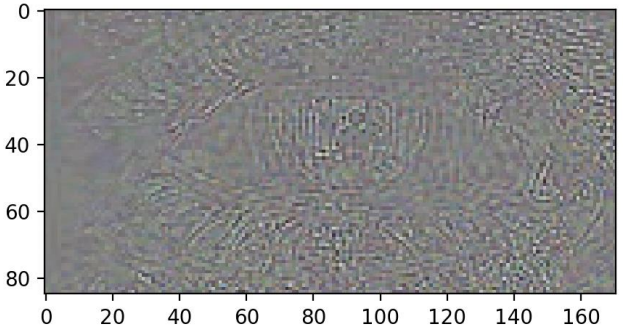
The JPEG Algorithm

- **Quantization matrix의 scale에 따른 결과 (block size : 30 x 30)**

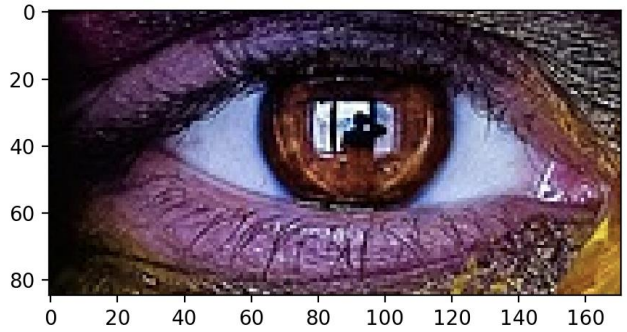
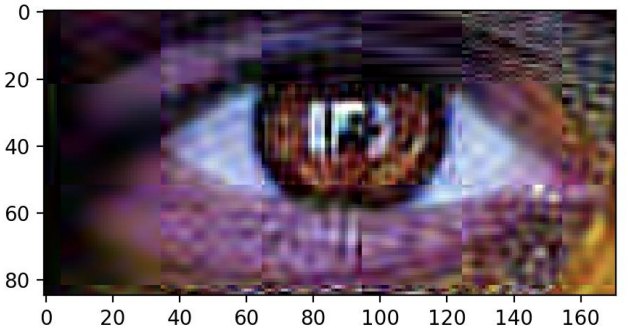
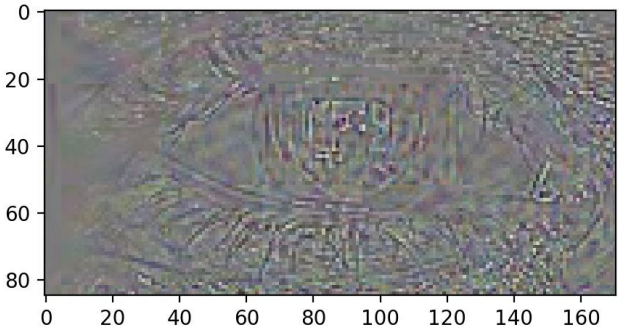
Scale : 1

Scale : 5

Scale : 10

과제 : JPEG

- **JPEG 구현**
  - JPEG 알고리즘의 encoding과 decoding을 구현

Encoding

Decoding

https://pixabay.com/photos/girl-face-colorful-colors-artistic-2696947/

과제 : JPEG

- **세부 사항**
  - 과제에서는 임의의 block size에 대해서도 동작할 수 있도록 구현
    - 이론에서는 8 x 8을 사용
  - 컬러 이미지에 대해 처리해야함
    - 각 채널에 대해서 encoding 및 decoding을 수행
    - 이후 채널 방향으로 쌓아 다시 컬러 이미지로 변환
  - 모든 파일에는 테스트 코드가 존재
    - Encoding 및 decoding은 오래걸리므로 테스트를 통해 디버깅하는 것을 권장

과제 : JPEG

- **dct.py**
  - Discrete cosine transform을 수행하는 클래스
  - Encoding에서는 이미지를 주파수의 형태로 변환해야함
    - spatial_to_frequency 메서드를 사용
  - Decoding에서는 주파수 형태의 값을 이미지로 변환해야함
    - frequency_to_spatial 메서드를 사용
  - 위 두 메서드에 대해 테스트 가능

```
------------------------------
[Discrete Cosine Transform]
Spatial domain --> frequency domain test... PASSED!
Frequency domain --> spatial domain test... PASSED!
------------------------------
```

과제 : JPEG

- **scanning.py**
  - Zigzag scanning 을 수행하는 클래스
  - Quantization이 완료된 값에 대해 scanning
    - encode 메서드를 사용
  - Decoding시, zigzag scanning를 역으로 수행해야함
    - decode 메서드를 사용
  - 위 두 메서드에 대해 테스트 가능

```
------------------------------
[Zigzag scanning]
Encoding test... PASSED!
Decoding test... PASSED!
------------------------------
```

과제 : JPEG

- **jpeg.py**
  - JPEG 알고리즘을 수행하는 클래스
  - 이미지를 블럭 단위로 쪼개거나, 블럭들을 다시 이미지로 만들어야함
    - image_to_blocks 및 blocks_to_image 메서드를 사용
  - Encoding 및 decoding시 블럭간의 잔차를 구하거나, 구해진 잔차로부터 원본을 구해야함
    - get_residual 및 get_inverse_residual 메서드를 사용
    - 두 메서드에 대해 테스트 가능
  - Encoding 및 decoding을 위한 메서드를 구현하는 것이 최종 목표

과제 : JPEG

- **main.py**
  - 이미지는 block size에 맞게 나누어 떨어지지 않을 수 있음
    - 이를 위해 이미지 오른쪽과 아래 부분에 대해 패딩을 적용
    - pad_right_and_bottom 함수 사용
    - 패딩 크기를 block size에 맞게 계산해야함

과제 : JPEG

- **과제 정리**
  - 각 파일의 비워진 메서드들을 구현
    - jpeg/dct.py
    - jpeg/scanning.py
    - jpeg/jpeg.py
  - main.py 부분의 비워진 메서드를 구현
    - pad_right_and_bottom
  - Block size는 본인이 원하는 크기로 설정
  - 결과 이미지는 35~37 페이지와 같이 출력되어야 함

과제

- **보고서**
  - 내용
    - 학과, 학번, 이름
    - 구현 코드: 구현한 코드에 대한 간단한 설명
    - 이미지: main.py 실행 시, 출력되는 이미지
    - 느낀 점: 구현 결과를 보고 느낀 점, 혹은 어려운 점 등
    - 과제 난이도: 개인적으로 느낀 난이도 및 이유(과제가 쉽다, 어렵다 등)
  - .pdf 파일로 제출(이외의 파일 형식일 경우 감점)
  - 보고서 명
    - [IP]20xxxxxxx_이름_12주차_과제.pdf

과제

- **과제 요약**
  - 채점 기준
    - 구현을 못하거나 잘못 구현한 경우
    - 보고서 내용이 빠진 경우
    - 다른 사람의 코드 copy 적발 시 보여준 사람, copy한 사람 둘 다 0점
    - **메서드 및 함수의 결과를 바로 생성할 수 있는 cv2및 numpy 함수는 사용금지**
  - 제출 파일

cv2 리시브인가 리사이즈
인가만 사용가능

    - 아래의 파일을 압축해서 [IP]20XXXXXXX_이름_12주차_과제.zip 으로 제출
      - .py 파일
        - » **python 파일들은 디렉토리 구조를 그대로 유지한 채로 제출해야함!**
      - .pdf 보고서 파일
  - 제출 기한
    - 00분반 : 2025년 6월 6일 08시 59분까지
    - 01분반 : 2025년 6월 6일 10시 59분까지

Q & A