CV DEEP LEARNING WITH PYTORCH

PART I

ABOUT CV

ABOUT COMPUTER VISION

◆ COMPUTER VISION

디지털 이미지나 비디오에서 **정보 추출/해석**하여 특정 **작업 수행**하는 **컴퓨터 과학 분야**

적용 분야

- 이미지 개선 및 변형 작업 => 이미지 크기 조정, 색상 보정, 노이즈 제거
- 패턴 인식 작업 => 특정 패턴/특징 검출 즉, 얼굴/문자 인식 기술
- 객체 인식 및 추척 => 이미지/비디오 속 특정 객체 식별/추적
- 객체 및 영역 분할 => 픽셀들 분할하는 기술

※ 자율주행자동차, 보안 시스템, 의료 진단 등 분야에서 핵심적 역할

ABOUT COMPUTER VISION

◆ COMPUTER VISION

STEP 1 : 카메라나 기타 센서에서 이미지 또는 비디오 획득 ➔ 픽셀 데이터

STEP 2 : 노이즈 및 아틱팩트(부산물) 제거 / 크기 조정 / 밝기 및 대비 조정

STEP 3 : 특징 추출

▶ 단순한 특징 : 가장자리나 모서리 ▶ 복잡한 특징 : 물체나 물체 일부

STEP 4 : 특징 사용하여 물체 식별 즉, 객체 인식(Object Detection)

▶ ML/탬플릿 매칭 등 다양한 기술 사용 수행

▶ 객체 식별로 장면 이해 가능

ABOUT COMPUTER VISION

◆ DISIGITAL IMAGE PROCESSING

- 디지털 이미지 신호 처리 분야
- **이미지로부터 유의미한 정보 얻기 위해 사용되는 알고리즘 의미**
- 변환/분류/탐지/인식/검출/분석/왜곡/수정/향상/복원/압축/필터링 등 다양한 처리
- 디지털 이미지 ▶ ▶ ▶ 2차원 배열 형태 표시 ▶ ▶ ▶ numpy의 ndarray 객체

ABOUT COMPUTER VISION

◆ DISIGITAL IMAGE PROCESSING

❖ **전처리 알고리즘**

▪ **불필요한 데이터 줄이고 유의미한 데이터로 정제하는 과정**

예) 객체 위치 탐지 시 객체 제외 모든 요소 제거

▪ 종류

- ➢ 데이터 폭 및 범위 **조절 알고리즘**
- ➢ 특정 지점 기준으로 데이터 **가공 알고리즘**
- ➢ 검출 쉽도록 **변형 알고리즘**

ABOUT COMPUTER VISION

◆ DISIGITAL IMAGE PROCESSING

❖ **노이즈(Noise) 및 디노이즈(Denoise)**

- 알고리즘 상에서 **원하지 않는 데이터**
- **다른 데이터를 간섭, 왜곡시키는 데이터**
- 원인 : 장면 전환, 피사체의 움직임, 카메라 성능 등 다양한 이유로 발생

- 처리 방법
  - ➢ 이미지를 번지게 하는 **블러링(Blurring)**
  - ➢ 임의이 필터 적용으로 **선명화(Sharpening)**
  - ➢ 특정 영역 내의 데이터 비교해서 **대체하는 작성** 등등

ABOUT COMPUTER VISION

◆ DISIGITAL IMAGE PROCESSING

❖ **특징 및 유사성 검출(Feature and Similarity Detection)**

- **이미지 내의 주요한 특징점**을 검출하는 방법
- 이미지 내의 **특이한 패턴이나 구조**를 찾아내는 것이 중요
- 특징점 존재 **위치, 특징점 부각 ,** 이미지 내의 **주요한 유사 영역 검출**

- 처리 방법
  - ➢ 경계선 인지 엣지 추출(edge detection)
  - ➢ 엣지 교차되는 코너 검출(Corner detection)
  - ➢ 검출된 엣지로부터 직선 부분 검출 (Line detection)

ABOUT COMPUTER VISION

◆ DISIGITAL IMAGE PROCESSING

❖ 특징 매칭(Feature Matching)

- 이미지에서 추출한 **특징 디스크립터**(Feature Descriptor) **비교하여 대응 특징점 쌍 검출**
- 이미지 검색, 객체 인식, 영상 분류 등의 분야에서 활용
- multi-view stereo 처럼 여러 장의 서로 다른 시점의 이미지들에서 대응점 검출 활용

- 처리 방법
  - ➢ 최근접 이웃(Nearest Neighbor) 매칭 : 거리 계산하여 가장 가까운 특징 매칭 방법
  - ➢ RANSAC(Random Sample Consensus) 매칭 : 잘못된 매칭을 걸러내기 위한 방법

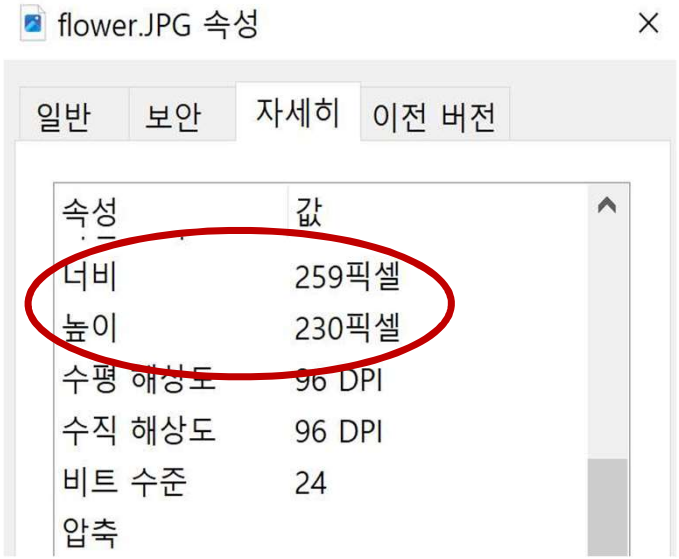
PART I

ABOUT IMAGE

ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **너비(Width/Column/X) & 높이(Hight/Row/Y)**



ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **너비(Width/Column/X) & 높이(Hight/Row/Y)**

- 행렬로 구성되며 이미지 크기 의미
- 이미지 크기만큼 데이터 생성
- 많은 알고리즘에서 전처리에 크기 변경 적용으로 데이터량 조절

```
00 00 08 03 00 00 27 10 00 00 00 1C 00 00 00 1C
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 54 B9 9F 97 3C 24
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 DE FE FE FE FE F1 C6 C6 C6 C6
```

ABOUT IMAGE

◆ IMAGE

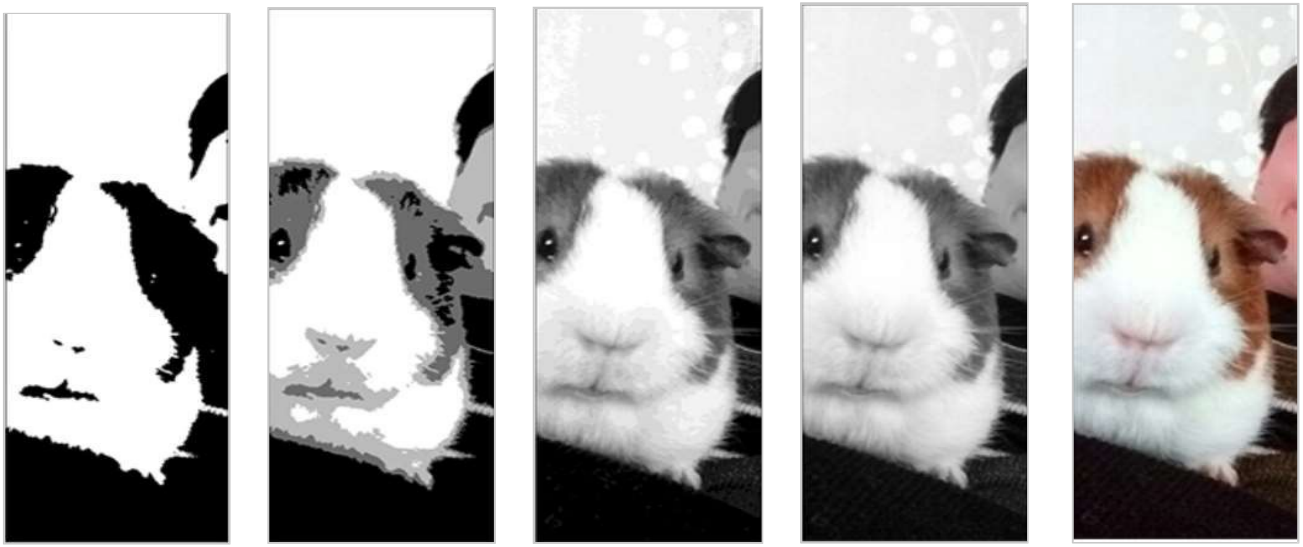
❖ 구성요소 - **정밀도(Bit Depth)**

- 이미지가 **얼마나 많은 색상 표현할 수 있는지 의미**
- **비트 깊이, 색상 깊이, 색 심도**라도고 함
  - ➢ 정밀도 높을 수록 ➔ 많은 색상 표현, 자연스러운 이미지
  - ➢ 정밀도 낮을 수록 ➔ 육한 확인 할 수 없을 정도 변형
- 표현 : nBit
- **최소 8Bit여야 유의미 데이터로 색상 표현 : 그레이스케일(Grayscale)**

ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **정밀도(Bit Depth)**



-1Bit- 이진화

-2Bit- 저화질

-4Bit- 저화질

-8bit- 그레이스케일

-32Bit- 컬러

ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **정밀도(Bit Depth)**

OpenCV 형식	데이터 형식	의미
np.uint8	byte	8-bit unsigned integers
np.int8	sbyte	8-bit signed integers
np.uint16	uint16	16-bit unsigned integers
np.int16	int16	16-bit signed integers
np.float32	float	32-bit floating point number
np.double	double	64-bit floating point number

ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **채널(Channel)**

- **이미지의 색상 정보** 담고 있는 요소
- **Red, Green, Blue**, Alpha, Hue, Saturation, Value
- 색상 표시

  ▶다중/다채널 : 3 ~ 4개 채널, 컬러이미지 ▶단일채널 : 1개 채널, 흑백이미지

- 많은 데이터량, 연산량 증가 ▶ ▶ ▶ 단일 채널 계산 진행
- 다중 채널의 불필요한 데이터 할당 ▶ ▶ ▶ 단일 채널 계산 진행으로 정확도 높임

ABOUT IMAGE

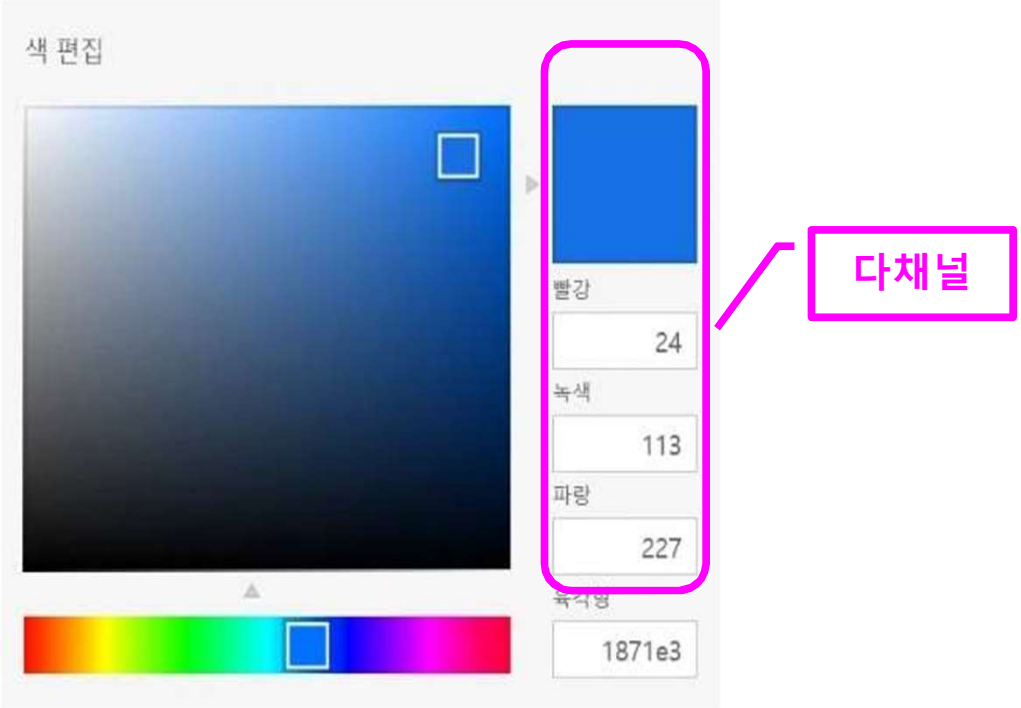
◆ IMAGE

❖ 구성요소 - **채널(Channel)**

ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **채널(Channel)**



ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **색공간(Color Space)**

- 색 표시계(color system)를 3차원으로 표현한 공간 개념
- 색 표시계의 모든 색들은 색 공간에서 3차원 좌표로 표현
- 색상(hue), 명도(lightness), 채도(chroma)를 3차원 공간 각각의 기본 축 공간

RGB 색 공간	CMYK 색 공간	HSV 색 공간
- 가산 혼합 방식 ➔ 명도 ▲ - Red, Green, Blue + Alpha - 활용 : Web 색상표현	- 감산 혼합 방식 ➔ 명도 ▼ - Cyan, Magenta, Yellow, Black - 활용 : 인쇄물	- 색상, 채도, 명도 직관적 - 활용 : 시각 예술

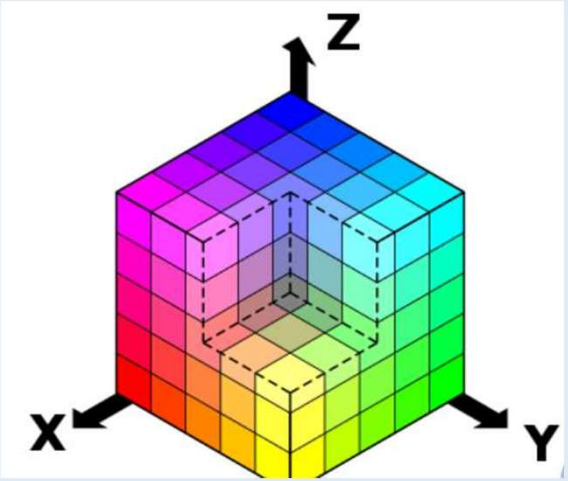
ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **색공간(Color Space)**

▪ RGB 색공간

- 방식 : 가산 혼합 방식 ➔ 명도 ▲
- 구성 : Red, Green, Blue + Alpha
- 활용 : Web 색상표현



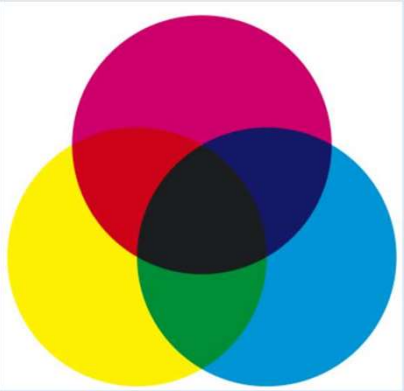
ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **색공간(Color Space)**

▪ CMYK 색공간

- 방식 : 감산 혼합 방식 ➔ 명도 ▼
- 구성 : 옥색(Cyan), 자홍색(Magenta), 노랑(Yellow), 검정(Black)
- 활용 : 인쇄



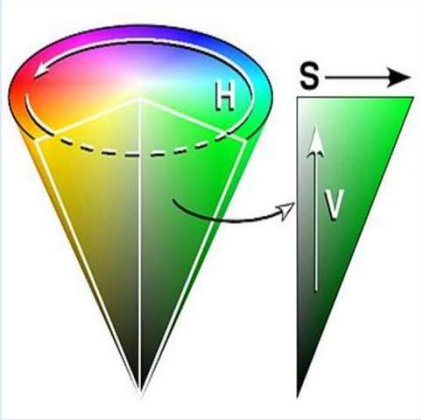
ABOUT IMAGE

◆ IMAGE

❖ 구성요소 - **색공간(Color Space)**

▪ HSV 색공간

- 방식 : 색상(Hue), 채도(Saturation), 명도(Value) 기준 , 직관적
- 구성 : 색상(Hue), 채도(Saturation), 명도(Value)
- 활용 : 시각예술



ABOUT IMAGE

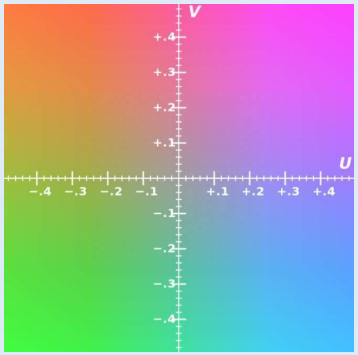
◆ IMAGE

❖ 구성요소 - **색공간(Color Space)**

▪ YUV 색공간

- 방식 : 휘도(밝기)기반 색상 분리
- 구성 : Y(Luminance 밝기) + U (Blue Color - 밝기) + V (Red Color - 밝기)
- 활용 : YCbCr (디지털 방식) , YPrPb(아날로그 방식)
- 특징 : RGB로 처리 시 데이터 너무 많음

$$Y = (0.257 \times R) + (0.504 \times G) + (0.098 \times B) + 16$$
$$U = -(0.148 \times R) - (0.291 \times G) + (0.439 \times B) + 128$$
$$V = (0.439 \times R) - (0.368 \times G) - (0.071 \times B) + 128$$



ABOUT IMAGE

◆ IMAGE

❖ 관심영역(ROI : Region Of Interest)

- 이미지 처리 시 **검출하고자하는 영역을 명확하게 지정한 것**
- 검출 제외한 주변 영역에 대한 불필요한 연산 발생 ➔ 연산량 많아짐
- **정확도와 연산 속도 향상** 위해 설정

ABOUT IMAGE

◆ IMAGE

❖ 관심채널(COI : Channel Of Interest)

- 이미지상에서 **관심 있는 채널 의미**
- 색상 데이터에는 많은 정보 포함
- 채널 분리해서 **특정 채널에 대해 연산 수행**으로 **데이터 양 1/3 줄어듬**
- 채널 모두 분리 뒤 동일 알고리즘 적용해 **더 많은 결과 얻는 방법** 사용
- **연산 1/3감소,획득 정보량 3배**

ABOUT IMAGE

◆ IMAGE

❖ 관심채널(COI : Channel Of Interest)

- 채널 분리 시 그레이스케일과 비슷한 형태
- Grascale 공식 : Y = **0.299 x R** + **0.587 x G** + **0.114 x B**

❖ **Green** 채널에 많은 영향 / Blue 채널에 가장 적은 영향

ABOUT IMAGE

◆ IMAGE

❖ 히스토그램

- 이미지의 **픽셀값에 대한 분포**를 히스토그램으로 표현
- 이미지의 **밝은 픽셀과 어두운 픽셀의 분포** 확인
- 이미지 일정 부분과 다른 부분의 **밝기 강약 차이 즉, 콘트라스트** 확인
- **이미지 색 분포, 객체 주변 픽셀 분포** 등 다양한 형태로 **해석/활용**

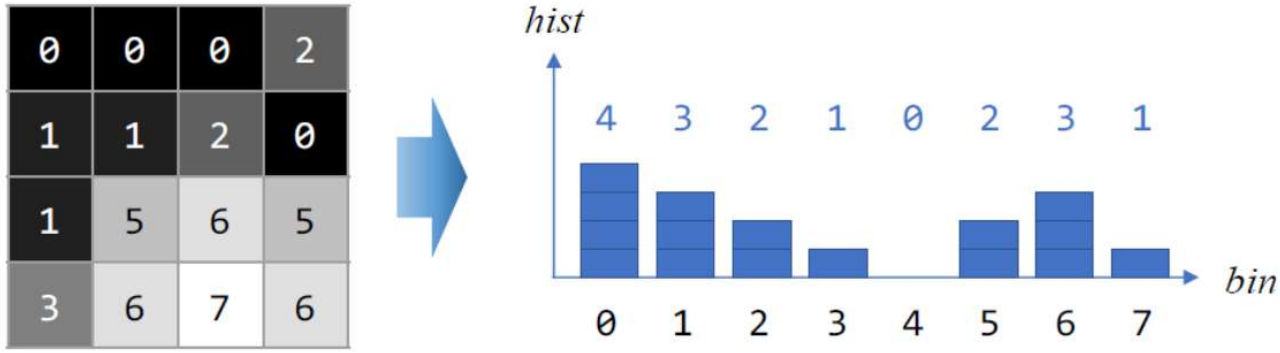
ABOUT IMAGE

◆ IMAGE

❖ 히스토그램

▪ 구성요소

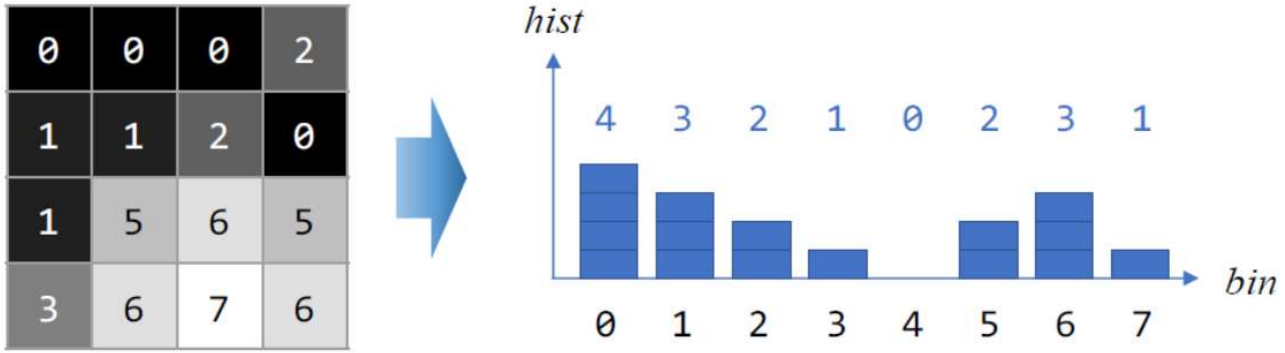
- 빈도수(BINS) : X축 간격, 256개 범위를 나눈 간격의 막대 수
- 차원수(DIMS) : 이미지 차원, 그레이스케일 1차원, 색상 이미지는 3차원 이상
- 범위(RANGE) : X축 범위, 0 ~ 255로 총 256개 범위



ABOUT IMAGE

◆ IMAGE

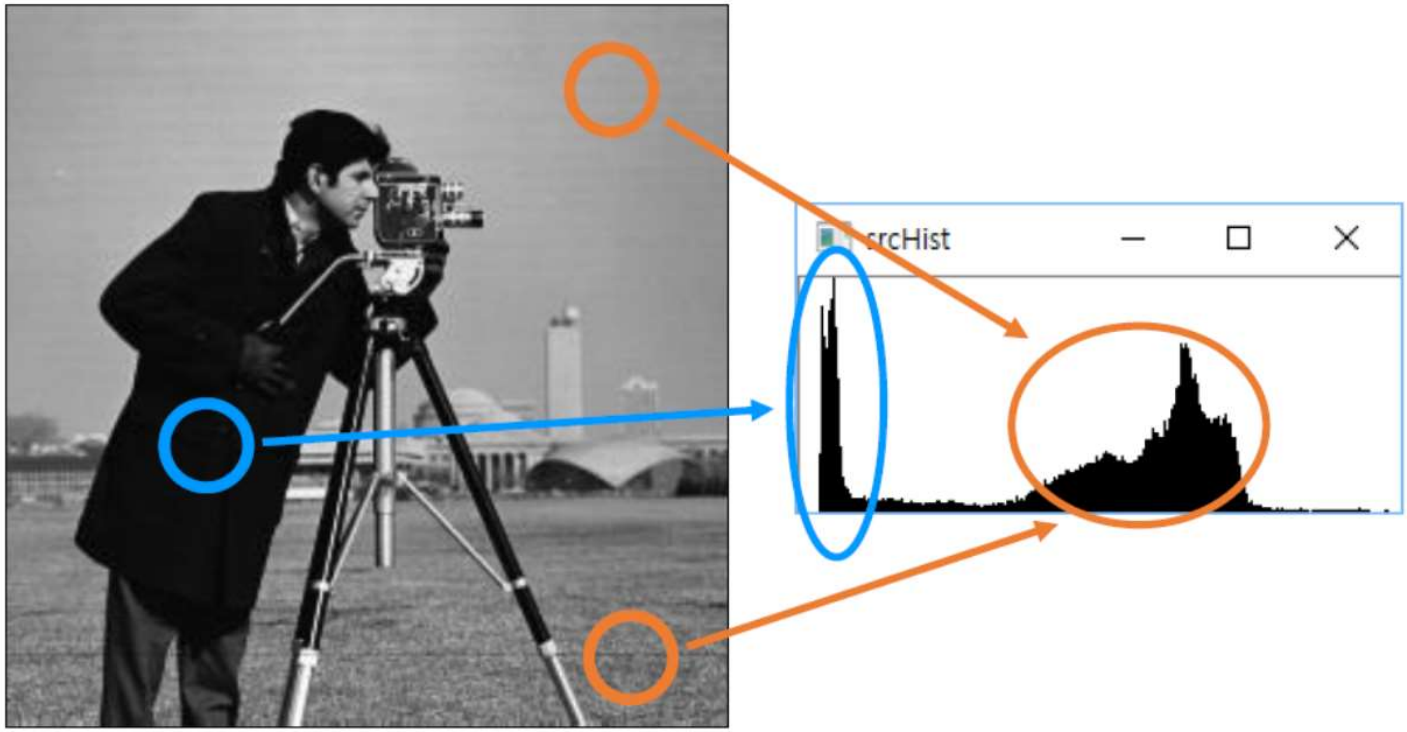
❖ 히스토그램



ABOUT IMAGE

◆ IMAGE

❖ 히스토그램



ABOUT IMAGE

◆ IMAGE

❖ 히스토그램

▪ 명암대비/콘트라스트/Contrast

ABOUT IMAGE

◆ MNIST Database

- 손으로 쓴 숫자들로 이루어진 대형 데이터베이스
- 다양한 화상 처리 시스템을 트레이닝하기 위해 일반적으로 사용
- 데이터베이스는 또한 기계 학습 분야의 트레이닝 및 테스트에 널리 사용
- 60,000개의 트레이닝 이미지와 10,000개의 테스트 이미지를 포함

ABOUT IMAGE

◆ MNIST Database

- http://yann.lecun.com/exdb/mnist/

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending mi formatting.

Four files are available on this site:

train-images-idx3-ubyte.gz: training set images (9912422 bytes)
train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

ABOUT IMAGE

◆ MNIST Database

- 데이터 파일

파일명	데이터
train-images-idx3-ubyte	학습 전용 6만개 이미지 정보 데이터 파일
train-labels-idx1-ubyte	6만개 이미지의 숫자 데이터 파일
t10k-images-idx3-ubyte	테스트 전용 1만개 이미지 정보 데이터 파일
t10k-labels-idx1-ubyte	1만개 이미지의 숫자 데이터 파일

ABOUT IMAGE

◆ MNIST Database

- 자체 데이터베이스 형식 - binary 타입

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

ABOUT IMAGE

◆ MNIST Database

- 자체 데이터베이스 형식 - binary 타입

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

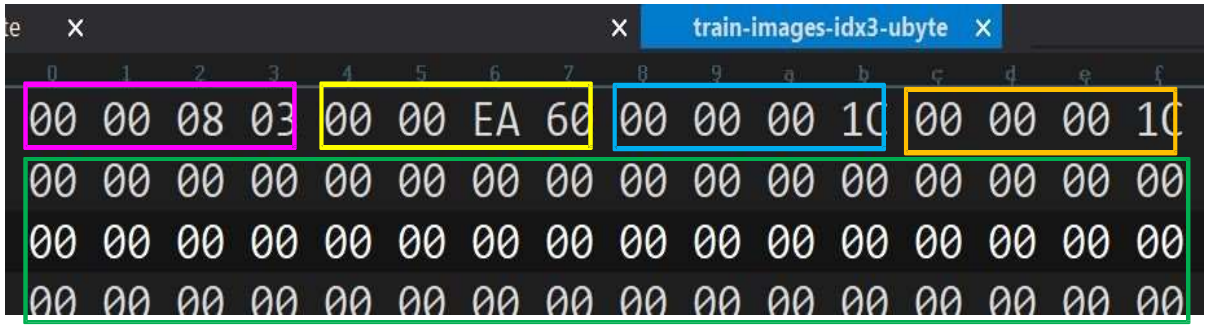
ABOUT IMAGE

◆ MNIST Database

- 자체 데이터베이스 형식 - binary 타입

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

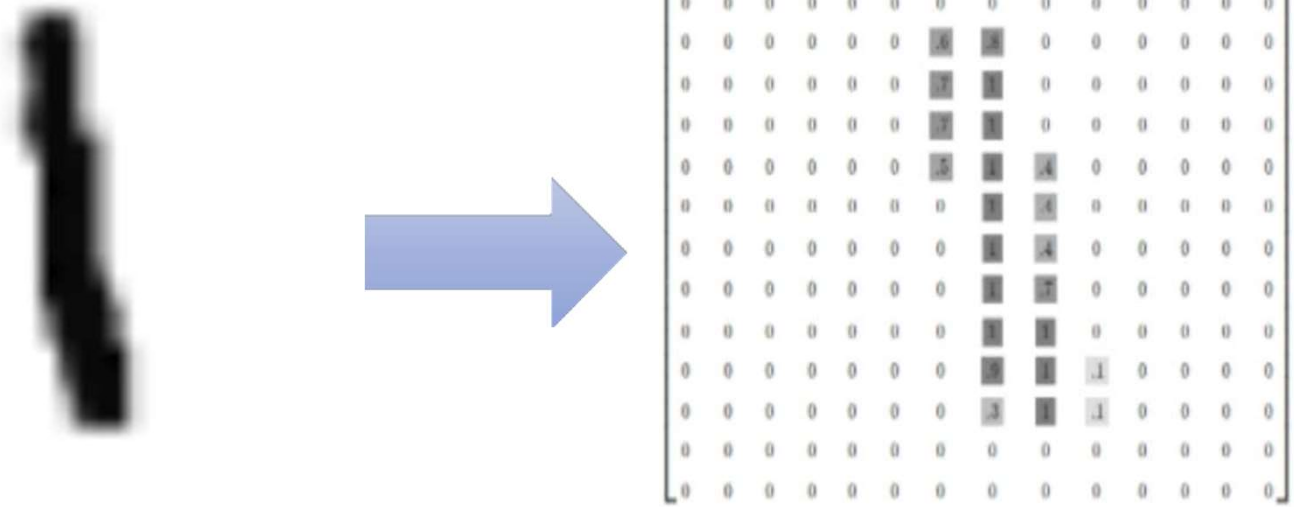
[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel



ABOUT IMAGE

◆ MNIST Database

- 이미지 데이터



ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

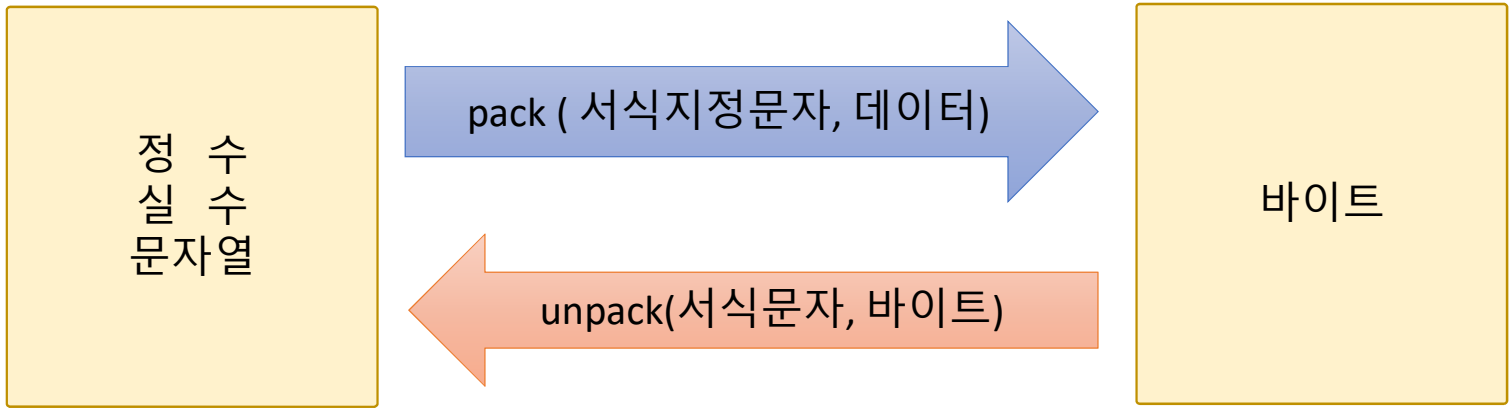
- C언어의 struct 구현한 모듈
- 바이너리 데이터 처리 모듈
- https://docs.python.org/3/library/struct.html?highlight=struct#module-struct

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

- 정수, 부동소수점, 문자열 ➔ bytes 객체로 변환



ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

- 포멧 문자열

Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		
c	char	bytes of length 1	1	
b	signed char	integer	1	(1), (2)
B	unsigned char	integer	1	(2)
?	_Bool	bool	1	(1)
h	short	integer	2	(2)
H	unsigned short	integer	2	(2)
i	int	integer	4	(2)
I	unsigned int	integer	4	(2)
l	long	integer	4	(2)
L	unsigned long	integer	4	(2)
q	long long	integer	8	(2)
Q	unsigned long long	integer	8	(2)
n	ssize_t	integer		(3)

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

- Endian - Byte Order : 메모리 공간에 여러 개의 연속된 바이트 데이터 배열하는 방법
  - Big Endian : 낮은 순서부터 저장 즉, 평소 숫자 읽는 순서대로 저장
  - Little Endian : 낮은 주소에 낮은 바이트 저장, 평소 숫자 읽는 반대로 저장

메모리주소	[빅엔디언]
addr 103	0x78
addr 102	0x56
addr 101	0x34
addr 100	**0x12**

데이터
0x12345678

메모리주소	[리틀엔디언]
addr 103	**0x12**
addr 102	0x34
addr 101	0x56
addr 100	0x78

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

- Byte Order : 1차원 메모리 공간에 여러 개의 연속된 데이터 배열하는 방법

Character	Byte order	Size	Alignment
@	native	native	native
=	native	standard	none
<	little-endian	standard	none
>	big-endian	standard	none
!	network (= big-endian)	standard	none

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

- Byte Order : 1차원 메모리 공간에 여러 개의 연속된 데이터 배열하는 방법

Character	Byte order	Size	Alignment
@	native	native	native
=	native	standard	none
<	little-endian	standard	none
>	big-endian	standard	none
!	network (= big-endian)	standard	none

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

```
import os, struct

# CSV 변환 기능 함수
def to_csv(name, maxdata):
    # (1) CSV 저장할 데이터 준비
    # 레이블 파일과 이미지 파일 열기
    lbl_f = open("../data/MNIST/"+name+"-labels-idx1-ubyte", "rb")
    img_f = open("../sata/MNIST/"+name+"-images-idx3-ubyte", "rb")

    # CSV 파일 생성
    csv_f = open("../data/MNIST/"+name+".csv", "w", encoding="utf-8")
```

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

```
# 헤더 정보 읽기
mag, lbl_count = struct.unpack(">II", lbl_f.read(8)) # 매직 코드 + 레이블 갯수
mag, img_count = struct.unpack(">II", img_f.read(8)) # 매직 코드 + 이미지 갯수
rows, cols = struct.unpack(">II", img_f.read(8))     # 행, 열 갯수
pixels = rows * cols

if DEBUG:
   print('lbl_count {}, img_count {}'.format(lbl_count, img_count))
   print('rows {}, cols {}'.format(rows, cols))
```

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

```
    # (2) 이미지 데이터를 읽고 CSV로 저장
    res = []
    for idx in range(lbl_count):
        if idx > maxdata: break

        # 숫자이미디 데이터가 의미하는 숫자값 읽기
        # 튜플타입 리던 ->1개 데이터 (value,)
        label = struct.unpack("B", lbl_f.read(1))[0]
        if DEBUG: print(' label = >{}'.format(label))
```

ABOUT IMAGE

◆ MNIST Database

❖ LIB - struct 모듈

```
    # 이미지 데이터 읽기
    bdata = img_f.read(pixels)
    sdata = list(map(lambda n: str(n), bdata))  # 문자열로 변환
    if DEBUG: print('sdata => {}'.format(sdata))

    # CSV 파일에 쓰기
    csv_f.write(str(label)+",")            # 숫자 라벨 쓰기
    csv_f.write(','.join(sdata) + "\r\n")  # 리스트 이미지 데이터->문자열 변환 쓰기
csv_f.close()
lbl_f.close()
img_f.close()
```

ABOUT IMAGE

◆ MNIST Database

❖ LIB - os 모듈

- 경로, 파일, 폴더 등등 운영체제 시스템, 파일 시스템 관련 함수, 클래스 제공

os.path.split(filename)	디렉토리와 파일명 분리
os.path.splitext(filename)	확장자만 분리
os.path.dirname(filename)	디렉토리만 분리
os.path.basename(filename)	파일이름만 분리
os.path.isdir (filename)	파일/ 디렉토리 여부 확인
os.path.join(dir, filename)	풀 경로 만들기
os.path.exists(dir-filename)	존재 여부 확인

ABOUT IMAGE

◆ MNIST Database

❖ LIB - os 모듈

- 폴더 내 모든 폴더 및 파일 리스트 추출

```
data_path='../data/img/‘          # 데이터 폴더경로

# 폴더 내 모든 폴더, 파일 추출
for dir_name in os.listdir(data_path):
    sub_path=(data_path+dir_name).lower()

    if os.path.isdir(sub_path):
       filelist=os.listdir(sub_path)
       print(f'--- [{dir_name}] filelist : {filelist}')
```

ABOUT IMAGE

◆ MNIST Database

❖ LIB - os 모듈

- 폴더 내 특정 확장자 파일 리스트 추출

```
data_path='../data/img/‘          # 데이터 폴더경로
file_ext='.png‘                   # 특정 추출 확장자


# 폴더 내 모든 폴더, 파일 추출
for dir_name in os.listdir(data_path):
    sub_path=(data_path+dir_name).lower()

    # 확장자 체크
    if sub_path.endswith(file_ext) :
       print(f'--- [{dir_name}] filelist : {sub_path}')
```

PART Ⅱ

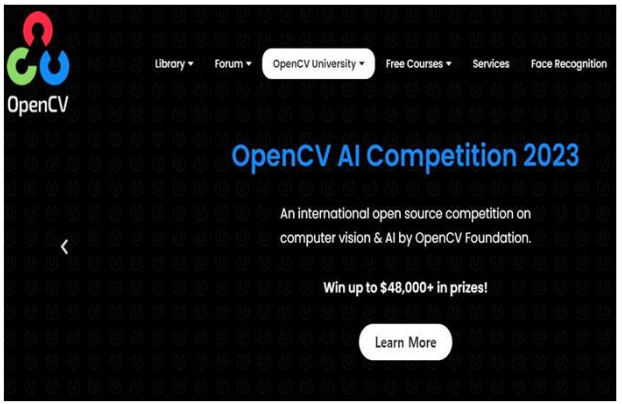
ABOUT OPENCV

ABOUT OPENCV

◆ OPENCV

- Open Source Computer Vision Library 약자
- **인텔에서 최초 개발한 Cross-Platform 기반 라이브러리**
- 실시간 영상 처리에 중점, 학술 및 상업적 용도 사용 가능
- **500가지가 넘는 알고리즘으로 최적화**
- 알고리즘 지원 함수는 알고리즘 수의 10배가 넘음
- **GPU 가속 모듈 지원으로 고해상도 이미지 실시간 처리 가능**
- **머신러닝과 밀접한 연관으로 머신러닝 관련 모듈 포함**
- https://opencv.org/

• https://opencv.org/



ABOUT OPENCV

◆ OPENCV 설치

❖ 가상환경 기반 권장

```
(AI_PY39) PS C:₩Users₩kwon> conda install -c conda-forge opencv=4.5.0
```

```
(AI_PY39) PS C:₩Users₩kwon> python
Python 3.9.17 (main, Jul  5 2023, 21:22:06) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2._ _version_ _
'4.5.0'
```