CV DEEP LEARNING WITH PYTORCH

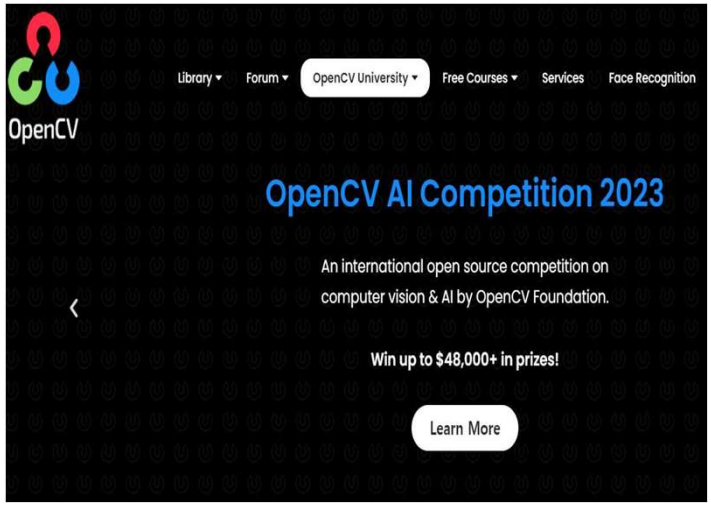
PART Ⅱ

ABOUT OPENCV

ABOUT OPENCV

◆ OPENCV

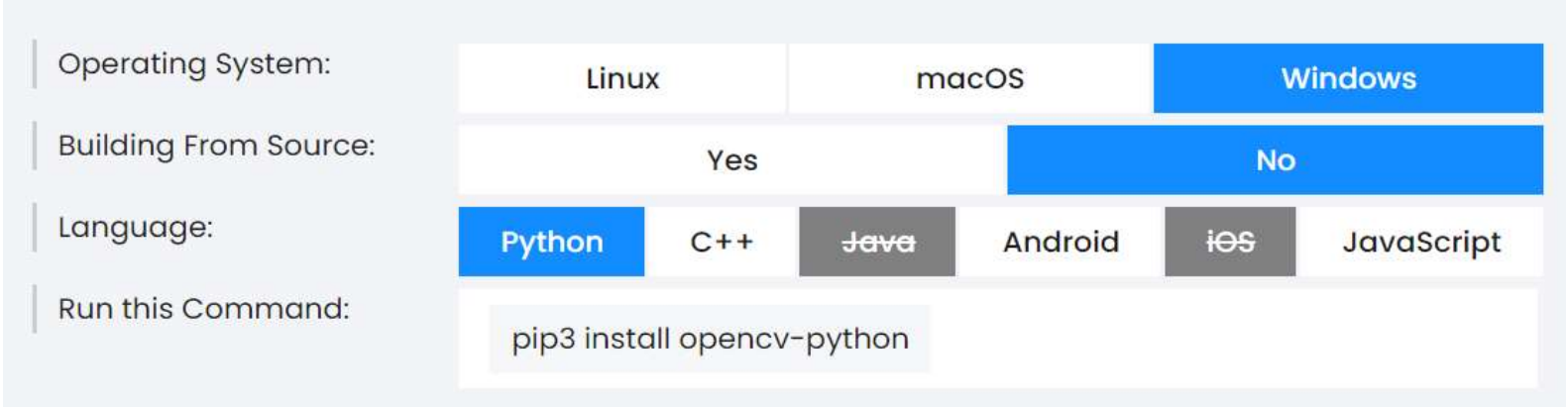
- Open Source Computer Vision Library 약자
- **인텔에서 최초 개발한 Cross-Platform 기반 라이브러리**
- 실시간 영상 처리에 중점, 학술 및 상업적 용도 사용 가능
- **500가지가 넘는 알고리즘으로 최적화**
- 알고리즘 지원 함수는 알고리즘 수의 10배가 넘음
- **GPU 가속 모듈 지원으로 고해상도 이미지 실시간 처리 가능**
- **머신러닝과 밀접한 연관으로 머신러닝 관련 모듈 포함**
- https://opencv.org/



ABOUT OPENCV

◆ OPENCV 설치

❖ https://opencv.org/get-started/




```
>>> import cv2
>>> cv2._ _version_ _
```

기본 주요 함수

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ ndarray = cv2.imread(filepath, flags)

- 지정된 경로의 이미지 데이터를 ndarray로 읽어오기
- 알파채널(투명) 있는 경우 BGRA 방식 읽기 ➔ 값 255 - 흰색 0 - 검은색

- filepath : 로딩할 이미지 경로포함 파일명
- floags
  - cv2.IMREAD_COLOR : 1 컬러이미지 [기본값]
  - cv2.IMREAD_GRAYSCALE : 0 그레이스케일
  - cv2.IMREAD_UNCHANGED : -1 속성 그대로

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ 이미지 기본 정보 출력

```
import cv2

img_file = '../data/img/flower.JPG'      # 이미지 경로
img = cv2.imread(img_file)               # 이미지 데이터 로딩

# 로딩된 이미지 데이터 정보
print(f'[타  입] {type(img)}')
print(f'[형  태] {img.shape}  [차원] {img.ndim}D')
print(f'[데이터] {img.dtype}  [크기] {img.size}byte')
```

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ 이미지 기본 정보 출력

```
import cv2

img_file = '../data/img/flower.JPG'                    # 이미지 경로
img = cv2.imread(img_file, cv.IMREAD_GRAYSCALE)    # 흑백 데이터 로딩

# 로딩된 이미지 데이터 정보
print(f'[타  입] {type(img)}')
print(f'[형  태] {img.shape}  [차원] {img.ndim}D')
print(f'[데이터] {img.dtype}  [크기] {img.size}byte')
```

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ 이미지 기본 정보 출력

```
# 2가지 모드의 이미지 로딩
img_gray  = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
img_color = cv2.imread(img_file, cv2.IMREAD_COLOR)

# 흑백이미지
print(f'[IMG_GRAY]                    [타  입] {type(img_gray)}')
print(f'[형  태] {img_gray.shape}   [차원] {img_gray.ndim}D')
print(f'[데이터] {img_gray.dtype}   [크기] {img_gray.size}byte')

# 컬러 이미지
print(f'[IMG_COLOR]‘                  [타  입] {type(img_color)}')
print(f'[형  태] {img_color.shape}  [차원] {img_color.ndim}D')
print(f'[데이터] {img_color.dtype}  [크기] {img_color.size}byte')
```

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ ndarray = cv2.imshow(title , image)

- 이미지 **출력 창 제어 코드 없으면 바로 종료**
- cv2.waitKey(), cv2.destroyAllWindows()

- title : 이미지 출력 창 제목
- image : numpy.ndarray 데이터

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ retval=cv2.waitkey (delay = 0)

- 이미지 출력 창 위에 keyboard 입력 대기하는 함수

- delay : 지연 시간
  - ➢ 밀리세컨드 값 입력
  - ➢ 0 무한대기
- retval : 1byte 사용자 입력 키 값

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ retval=cv2.waitkey (delay = 0)

- 키 코드값
  - 특정 키 코드값 : ord(문자')
  - 주요 특수키 코드값
    - ESC키 - 27
    - ENTER키 - 13
    - TAB키 - 9

제어 문자		공백 문자		구두점		숫자		알파벳			
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ 이미지 화면 출력

```
import cv2

img_file = "../data/img/flower.JPG"          # 이미지 경로
img = cv2.imread(img_file)                   # 이미지 데이터 로딩

if img is not None:
    cv2.imshow('FLOWER', img)                # 이미지 창 화면 출력
    cv2.waitKey()                            # 이미지 창 위에 키 입력 대기
    cv2.destroyAllWindows()                  # 창 모두 닫기
else:
    print('No image file!!!')
```

OPENCV BASIC

◆ 이미지 데이터 읽기

❖ 이미지 화면 출력

```
if img is not None:
    cv2.imshow('IMG', img)
    while True:
        key=cv2.waitKey()                          # 키 입력 대기
        print(f'key =>> {key} - {hex(key)}')  # 입력 키 출력

        if key==27:                                # ESC키 일때 화면 종료
            cv2.destroyAllWindows()
            break
else:
    print('No image file!!!')
```

OPENCV BASIC

◆ 이미지 데이터 저장

❖ ret = cv2.imwrite(filename, image)

- 지정된 경로에 이미지 저장

- filepath : 저장될 이미지 경로포함 파일명
- ret : 저장 여부

OPENCV BASIC

◆ 이미지 데이터 저장

❖ 회색 이미지 저장

```
import cv2

img_file = '../data/img/flower.JPG'                    # 이미지 경로
img_gray  = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)  # 흑백이미지

save_filename='../data/img/gray_.jpg'
gray_file = cv2.imwrite(save_filename, img_gray)        # 이미지 저장
```

OPENCV BASIC

◆ 이미지 데이터 채널 분리

❖ multi_channel = cv2.split(image)

- 이미지의 색상 분리
- 분리된 채널은 단일 채널로 흑백 색상 표현
- **주의! OpenCV의 색상 배열 순서 ➔ BGR**

▪ image	: ndarray 이미지 데이터
▪ multi_channel	: 분리된 데이터 리스트 B, G, R 반환

OPENCV BASIC

◆ 이미지 데이터 채널 분리

❖ 이미지 채널 분리

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

img_b, img_g, img_r = cv2.split(img)         # 이미지 채널 분리 B G R

cv2.imshow('[Blue Channel]', img_b)
cv2.imshow('[Green Channel]', img_g)
cv2.imshow('[Red Channel]', img_r)

cv2.waitKey()
cv2.destroyAllWindows()
```

OPENCV BASIC

◆ 이미지 데이터 채널 분리

❖ 이미지 채널 분리

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

# ndarray 분리
b, g, r = img[:,:,0], img[:, :, 1], img[:,:, 2]

cv2.imshow('[IMG-R]', b)
cv2.imshow('[IMG-G]', g)
cv2.imshow('[IMG-B]', r)
cv2.waitKey()
cv2.destroyAllWindows()
```

OPENCV BASIC

◆ 이미지 데이터 채널 병합 / 변경

❖ ndarray = cv2.merge((b, g, r))

- 분리된 채널 병합 및 채널 순서 변환 후 이미지 데이터 반환

- mv : 채널 튜플
- ndarray : 병합된 이미지 데이터

OPENCV BASIC

◆ 이미지 데이터 채널 병합/변경

❖ 이미지 채널 병합

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

img_b, img_g, img_r = cv2.split(img)         # 이미지 채널 분리 B G R

img_rgb = cv2.merge((img_r, img_g, img_b))   # 채널 순서 변경

cv2.imshow('[RGB IMAGE]', img_rgb)

cv2.waitKey()
cv2.destroyAllWindows()
```

OPENCV BASIC

◆ 이미지 색 공간 변환

❖ ndarray = cv2.cvtColor(src, code, dst=None, dstCn=None)

- BGR색 공간을 HSV, YCrCb, GrayScale 등 다른 색 공간 변환

- src : 입력 이미지 데이터
- code : 색공간
  - cv2.COLOR_BGR2GRAY / cv2.COLOR_GRAY2BGR
  - cv2.COLOR_BGR2RGB / cv2.COLOR_RGB2BGR
  - cv2.COLOR_BGR2HSV / cv2.COLOR_HSV2BGR
  - cv2.COLOR_BGR2YCrCb / cv2.COLOR_YCrCb2BGR
- dst : 결과 이미지 데이터
- dstCn : 결과 영상 채널 수

OPENCV BASIC

◆ 이미지 색 공간 변환

❖ 이미지 채널 BGR ➔ HSV

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

# BGR -> HSV
src_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
planes = cv2.split(src_hsv)

cv2.imshow('[HSV IMG]', src_hsv)

cv2.waitKey()
cv2.destroyAllWindows()
```

OPENCV BASIC

◆ 이미지 크기 변경

❖ ndarray = cv2.resize(src, dstSize, fx, fy, interpolation)

- BGR색 공간을 HSV, YCrCb, GrayScale 등 다른 색 공간 변환

- src : 입력 이미지 데이터
- dstSize : 변경 이미지 **절대 크기 (너비, 높이)**
- fx, fy : 변경 이미지 상대 크기, 절대크기(0, 0) 설정 필요
- interpolation : 보간법, 변형된 이미지 픽셀 추정 값 할당 방법
  - cv2.INTER_NEAREST 이웃보간법
  - cv2.INTER_LINEAR 쌍 선형 보간법 [확대]
  - cv2.INTER_LINEAR_ EXACT 비트 쌍 선형 보간법
  - cv2.INTER_CUBIC 바이큐빅 보간법 [확대]
  - cv2.INTER_AREA 영역 보간법 [축소]
  - cv2.INTER_LANCZOS4 Lanczos 보간법

OPENCV BASIC

◆ 이미지 크기 변경

❖ 이미지 축소 / 확대

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

down_img = cv2.resize(img, dsize=(30, 30), interpolation=cv2.INTER_AREA)
up_img = cv2.resize(img, dsize=(0, 0), fx=1.5, fy=1.5, interpolation=cv2.INTER_CUBIC)

cv2.imshow( "down_img", down_img )
cv2.imshow( "up_img", up_img )
cv2.imshow('ORG', img)

cv2.waitKey()
cv2.destroyAllWindows()
```

OPENCV BASIC

◆ 이미지 크기 변경

❖ ndarray = cv2.resize(src, dstSize, fx, fy, interpolation)

- BGR색 공간을 HSV, YCrCb, GrayScale 등 다른 색 공간 변환

- src : 입력 이미지 데이터
- dstSize : 변경 이미지 **절대 크기 (너비, 높이)**
- fx, fy : 변경 이미지 상대 크기, 절대크기(0, 0) 설정 필요
- interpolation : 보간법, 변형된 이미지 픽셀 추정 값 할당 방법
  - cv2.INTER_NEAREST 이웃보간법
  - cv2.INTER_LINEAR 쌍 선형 보간법 [확대]
  - cv2.INTER_LINEAR_ EXACT 비트 쌍 선형 보간법
  - cv2.INTER_CUBIC 바이큐빅 보간법 [확대]
  - cv2.INTER_AREA 영역 보간법 [축소]
  - cv2.INTER_LANCZOS4 Lanczos 보간법

OPENCV BASIC

◆ 이미지 크기 변경

❖ 이미지 축소 / 확대

```
import cv2

img_file = '../data/img/flower.JPG'          # 이미지 경로
img  = cv2.imread(img_file)                  # 이미지 로딩

down_img = cv2.resize(img, dsize=(30, 30), interpolation=cv2.INTER_AREA)
up_img = cv2.resize(img, dsize=(0, 0), fx=1.5, fy=1.5, interpolation=cv2.INTER_CUBIC)

cv2.imshow( "down_img", down_img )
cv2.imshow( "up_img", up_img )
cv2.imshow('ORG', img)

cv2.waitKey()
cv2.destroyAllWindows()
```

DRAWING FUNCTION

DRAWING FUNCTION

◆ 선 그리기 함수

❖ cv2.line(img, start, end, color, thickness, lineType)

- img : 그림을 그릴 이미지 파일
- start : 선 시작 좌표(ex; (0,0))
- end : 선 종료 좌표(ex; (500. 500))
- color : BGR형태의 선 색상 (ex; (255, 0, 0) -> Blue)
- thickness (int) : 선의 두께. pixel (default=1)
- lineType : 선 그리기 형식 (cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA)

DRAWING FUNCTION

◆ 사각형 그리기 함수

❖ cv2.rectangle(img, start, end, color, thickness, lineType)

- img : 그림을 그릴 이미지 파일
- start : 시작 꼭지점 좌표(ex; (0,0))
- end : 종료 꼭지점 좌표(ex; (500. 500))
- color : BGR형태의 선 색상 (ex; (255, 0, 0) -> Blue)
- thickness (int) : 선의 두께. pixel (default=1)
- lineType : 선 그리기 형식 (cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA)

DRAWING FUNCTION

◆ 다각형 그리기 함수

❖ cv2.polylines(img, pts, isClosed, color, thickness, lineType)

- img : 그림을 그릴 이미지 파일
- pts : 연결할 꼭짓점 좌표, Numpy array
- isClosed : 닫힌 도형 여부, True/False
- color : BGR형태의 선 색상 (ex; (255, 0, 0) -> Blue)
- thickness (int) : 선의 두께. pixel (default=1)
- lineType : 선 그리기 형식 (cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA)

DRAWING FUNCTION

◆ 원 그리기 함수

❖ cv2.circle(img, center, radius, color, thickness, lineType)

- img : 그림을 그릴 이미지 파일
- center : 타원의 중심 좌표 (x, y)
- radius : 반지름
- color : BGR형태의 선 색상 (ex; (255, 0, 0) -> Blue)
- thickness (int) : 선의 두께. pixel (default=1)
- lineType : 선 그리기 형식 (-1: 안쪽 채움)

DRAWING FUNCTION

◆ 타원 그리기 함수

❖ **cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness, lineType)**

- img: 그림을 그릴 이미지 파일
- center: 타원의 중심 좌표 (x, y)
- axes: 타원의 중심에서 가장 긴 축의 길이와 가장 짧은 축의 길이
- angle: 타원의 기준 축 회전 각도
- startAngle: 타원의 호가 시작하는 각도
- endAngle: 타원의 호가 끝나는 각도
- thickness (int): 선의 두께. pixel (default=1)
- lineType: 선 그리기 형식 (-1 : 안쪽 채움)

DRAWING FUNCTION

◆ 글자 쓰기함수

❖ cv2.putText(img, text, org, font, fontScale, color, thickness, lineType)

- img: 그림을 그릴 이미지 파일
- text: 출력 문자열
- org: 문자열 표시할 위치(좌측 하단 기준), (x, y)
- font: 글꼴 cv2.FONT_XXX 형식
- fontScale: 글꼴 크기
- color: 글자 색
- thickness (int): 글자 두께. pixel (default=1)
- lineType: 글자 선 형식 (cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA)

DRAWING FUNCTION

◆ 도형 및 글자 그리기

```
import cv2
import numpy as np

src = np.zeros((768, 1366, 3), dtype=np.uint8)

src = cv2.line(src, (100, 100), (1200, 100), (0, 0, 255), 3, cv2.LINE_AA)
src = cv2.circle(src, (300, 300), 50, (0, 255, 0), cv2.FILLED, cv2.LINE_4)
src = cv2.rectangle(src, (500, 200), (1000, 400), (255, 0, 0), 5, cv2.LINE_8)
src = cv2.ellipse(src, (1200, 300), (100, 50), 0, 90, 180, (255, 255, 0), 2)
```

DRAWING FUNCTION

◆ 도형 및 글자 그리기

```
pts1 = np.array([[100, 500], [300, 500], [200, 600]])
pts2 = np.array([[600, 500], [800, 500], [700, 600]])
src = cv2.polylines(src, [pts1], True, (0, 255, 255), 2)
src = cv2.fillPoly(src, [pts2], (255, 0, 255), cv2.LINE_AA)

src = cv2.putText(src, "HAPPY", (900, 600),
                  cv2.FONT_HERSHEY_COMPLEX, 2, (255, 255, 255), 3)

cv2.imshow("src", src)
cv2.waitKey()
cv2.destroyAllWindows()
```

DRAWING FUNCTION

◆ 얼굴인식

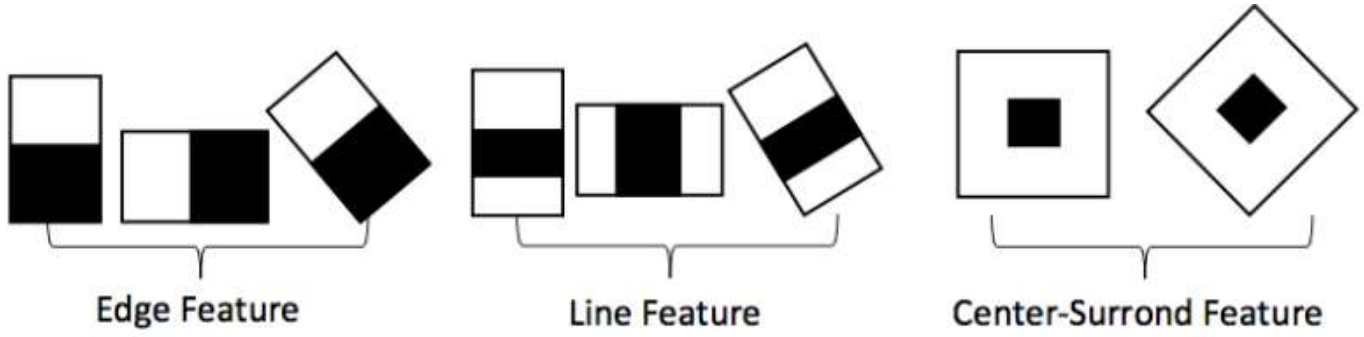
```
pts1 = np.array([[100, 500], [300, 500], [200, 600]])
pts2 = np.array([[600, 500], [800, 500], [700, 600]])
src = cv2.polylines(src, [pts1], True, (0, 255, 255), 2)
src = cv2.fillPoly(src, [pts2], (255, 0, 255), cv2.LINE_AA)

src = cv2.putText(src, "HAPPY", (900, 600),
                  cv2.FONT_HERSHEY_COMPLEX, 2, (255, 255, 255), 3)

cv2.imshow("src", src)
cv2.waitKey()
cv2.destroyAllWindows()
```

DRAWING FUNCTION

◆ 얼굴인식

❖ OpenCV 제공 인식 모델

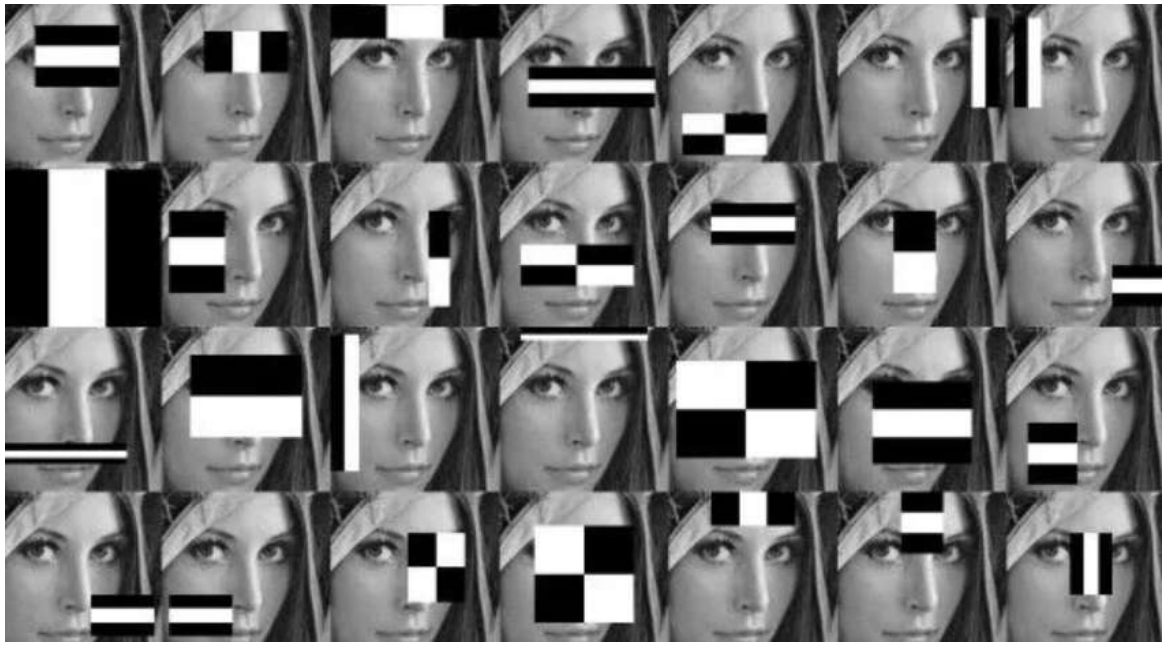
- Haar Cascade는 머신 러닝기반의 객체 검출 알고리즘
- 2001년 논문 "Rapid Object Detection using a Boosted Cascade of Simple Features"에서 Paul Viola와 Michael Jones가 제안한 특징(feature) 기반으로 비디오 또는 이미지에서 객체 검출
- 특징 : 직사각형 영역으로 구성되는 특징 사용으로 픽셀을 직접 사용할 때 보다 동작 속도 빠름



DRAWING FUNCTION

◆ 얼굴인식

❖ OpenCV 제공 인식 모델



DRAWING FUNCTION

◆ 얼굴인식

❖ OpenCV 제공 인식 모델

- haarcascades 폴더
  - haarcascade_eye.xml
  - haarcascade_eye_tree_eyeglasses.xml
  - haarcascade_frontalcatface.xml
  - haarcascade_frontalcatface_extended.xml
  - haarcascade_frontalface_alt.xml
  - haarcascade_frontalface_alt_tree.xml
  - haarcascade_frontalface_default.xml
  - haarcascade_fullbody.xml
  - haarcascade_lefteye_2splits.xml:
  - haarcascade_smile.xml
  - haarcascade_upperbody.xmlhaarcascade_smile.xml
  - haarcascade_upperbody.xml

DRAWING FUNCTION

◆ 얼굴인식

❖ **retval = cv2.CascadeClassifier.load(filename)**

객체 인식 모델 로딩

- filename : XML 모델 파일
- retval : 성공 True, 실패 False

DRAWING FUNCTION

◆ 얼굴인식

❖ **retval = cv2. CascadeClassifier.detectMultiScale(image, scaleFactor=None, minNeighbors=None, flags=None, minSize=None, maxSize=None)**

특정 영상에서 내가 찾고자 하는 객체를 검출

- image : 입력 영상 (cv2.CV_8U)
- scaleFactor : 영상 축소 비율. 기본값은 1.1.
- minNeighbors : 얼마나 많은 이웃 사각형 검출되어야 최종 검출 영역으로 설정할지 지정 기본값 3
- flags : (현재) 사용되지 않음
- minSize : 최소 객체 크기. (w, h) 튜플
- maxSize : 최대 객체 크기. (w, h) 튜플
- result : 검출된 객체의 사각형 정보(x, y, w, h) 담은 numpy.ndarray

DRAWING FUNCTION

◆ 얼굴인식

```
# 이미지 파일 및 정면얼굴 인식 모델 경로
frontalface_model = '../data/haarcascades/haarcascade_frontalface_alt.xml'
img_file = '../data/img/black_pink.jpg‘

# 정면얼굴 인식 모델 로딩
face_detector = cv2.CascadeClassifier(frontalface_model)

# 이미지 데이터 로딩
imgNP=cv2.imread(img_file)

# 이미지에서 정면 얼굴 인식 ==> [결과] 인식된 얼굴 수 만큼 위치정보(x, y, w, h) 반환
face_detections = face_detector.detectMultiScale(imgNP)
print(f'** face_detections RESULT₩n{face_detections}')
```

DRAWING FUNCTION

◆ 얼굴인식

```
# 인식된 정면 얼굴 ROI 표시
for (x, y, w, h) in face_detections:
    cv2.rectangle(imgNP, (x, y), (x + w, y + h), (0, 255, 0), 2)

# 화면 출력
cv2.imshow('[M]', imgNP)

# 키 입력 및 X버튼에 따른 창 닫기
if cv2.waitKey() or cv2.getWindowProperty('image', cv2.WND_PROP_VISIBLE) <1:
    cv2.destroyAllWindows()
```