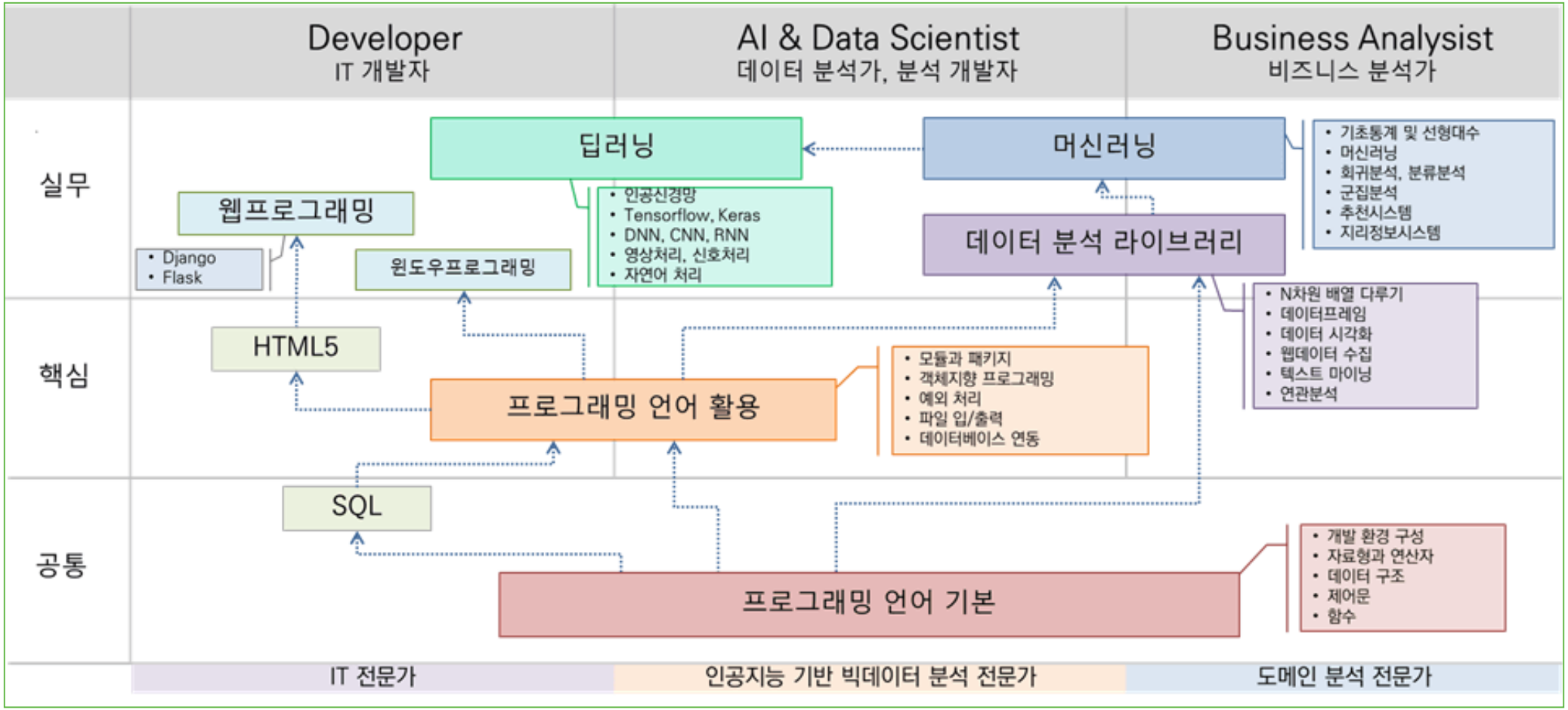
이미지 분류를 위한 인공지능
CNN 모델 원리와 구현

이 슬라이드에서 사용한 서체 :
- Open Sans(https://ko.cooltext.com/Download-Font-Open+Sans)
- KoPubWorld돋움체(http://www.kopus.org/biz/electronic/font.aspx)

인공지능 로드맵

이미지 분류를 위한 인공지능 CNN 모델 원리와 구현



실습 환경 (선택 1)

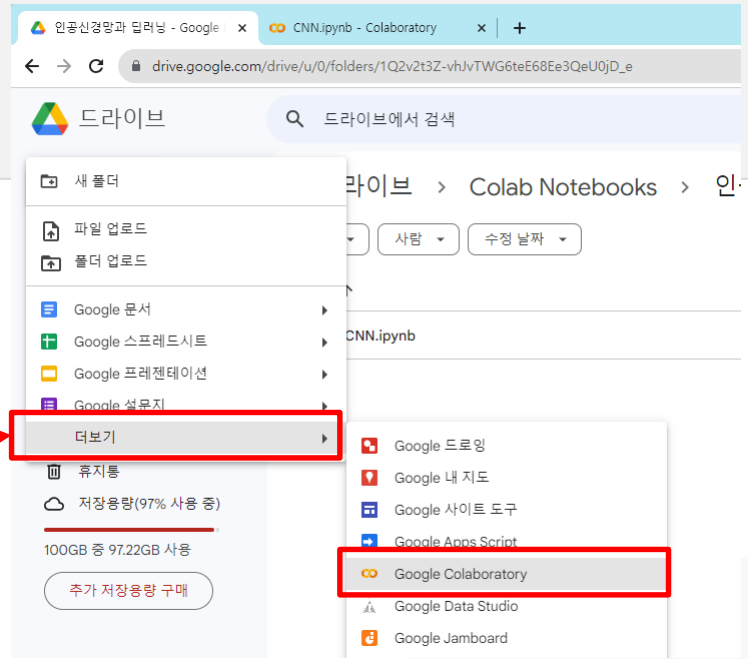
1. https://repo.anaconda.com/archive/에서 아나콘다 다운로드 및 설치
   - 강의장 운영체제는 Windows 10, GPU 2080
   - GPU를 사용하려면 Anaconda3-2023.03-1-Windows-x86_64.exe 파일을 내려받아 설치하세요.
   - 파이썬 개발 인터프리터, 주피터노트북 개발환경, 머신러닝 및 딥러닝을 위한 패키지 등이 설치됨

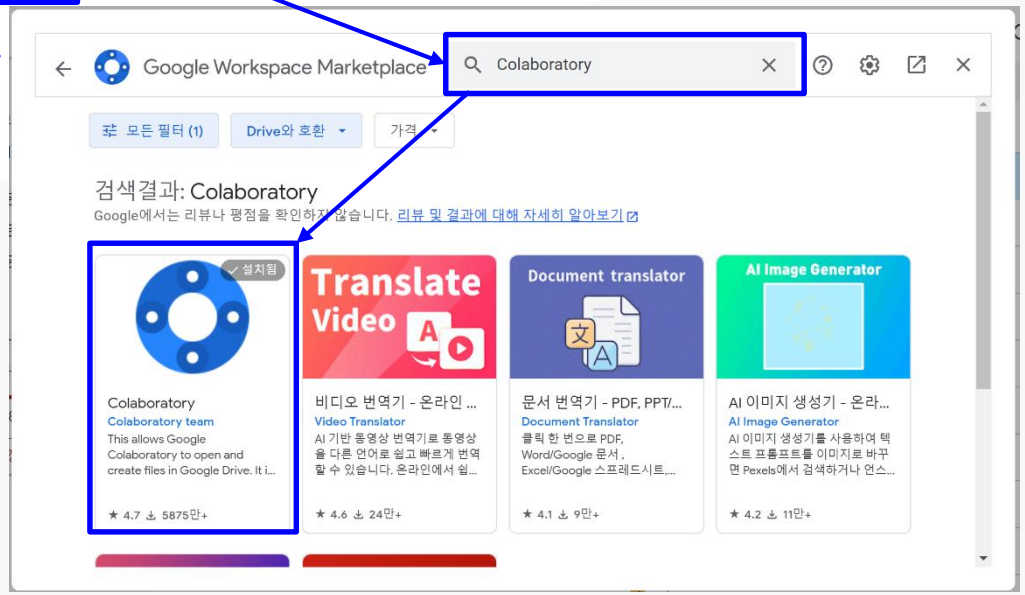
2. 아나콘다 설치 후 Anaconda prompt를 관리자 권한으로 실행하고 패키지 설치
   - 윈도우 10에서 GPU 사용: pip install tensorflow==2.10
   - 윈도우 11: pip install tensorflow 또는 pip install tensorflow-gpu

실습 환경 (선택 2)

이미지 분류를 위한 인공지능 CNN 모델 원리와 구현

1. 개발 및 실행 환경은 Google Colab notebook을 사용합니다.
    1. Chrome 브라우저에서 구글 계정으로 로그인
    2. 구글 드라이브로 이동(https://drive.google.com/)
    3. + 신규 -> 더보기 -> Google Colaboratory 선택
        - Colaboratory가 없을 경우 `연결할 앱 더보기`를 선택하고 Colaboratory를 검색하여 설치하세요.
    4. 코드 작성 후 실행은 '컨트롤 + 엔터'

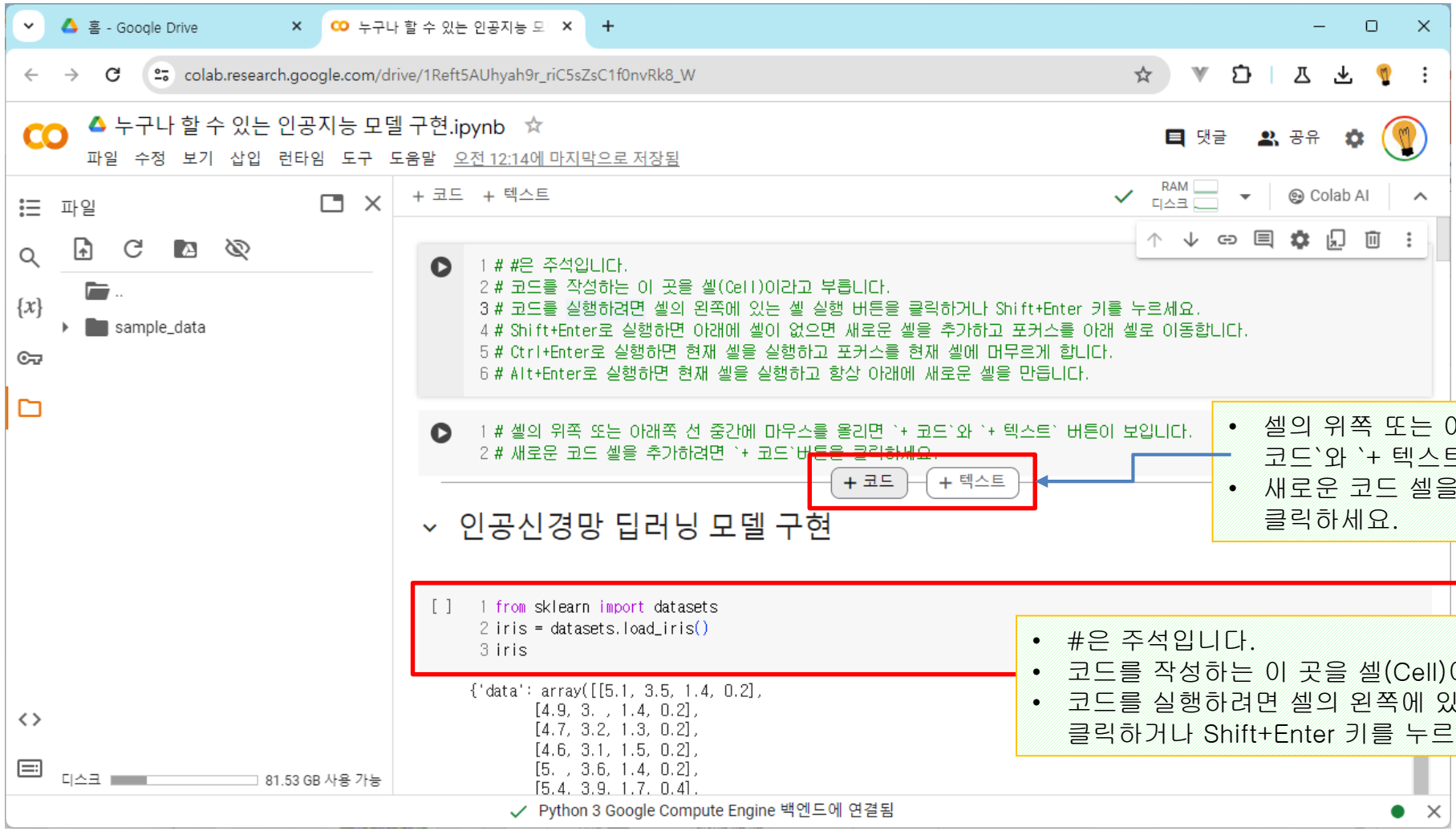





```
# 코랩에서 한글 그래프 적용하려면(세션 다시 시작)
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

코드 작성 및 실행

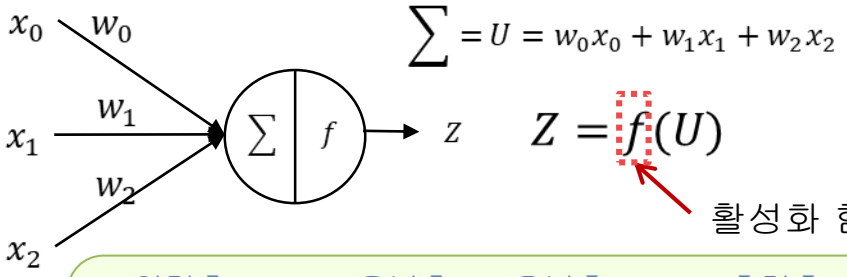
이미지 분류를 위한 인공지능 CNN 모델 원리와 구현



- Shift+Enter로 실행하면 아래에 셀이 없으면 새로운 셀을 추가하고 포커스를 아래 셀로 이동합니다.
- Ctrl+Enter로 실행하면 현재 셀을 실행하고 포커스를 현재 셀에 머무르게 합니다.
- Alt+Enter로 실행하면 현재 셀을 실행하고 항상 아래에 새로운 셀을 만듭니다.

- 셀의 위쪽 또는 아래쪽 선 중간에 마우스를 올리면 `+ 코드`와 `+ 텍스트` 버튼이 보입니다.
- 새로운 코드 셀을 추가하려면 `+ 코드`버튼을 클릭하세요.

- #은 주석입니다.
- 코드를 작성하는 이 곳을 셀(Cell)이라고 부릅니다.
- 코드를 실행하려면 셀의 왼쪽에 있는 셀 실행 버튼을 클릭하거나 Shift+Enter 키를 누르세요.

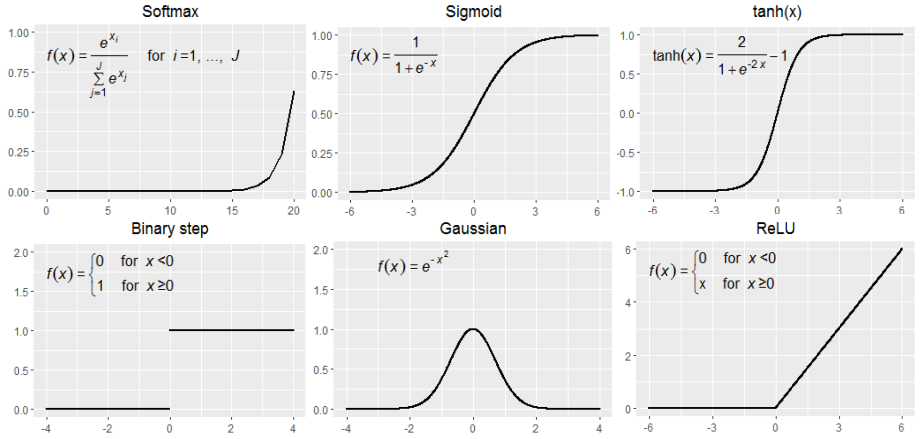


$$\sum = U = w_0x_0 + w_1x_1 + w_2x_2$$

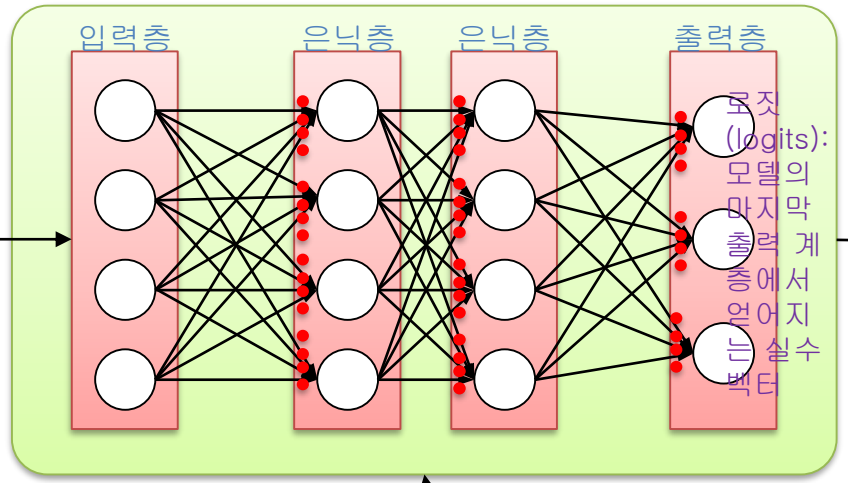
$$Z = f(U)$$

활성화 함수

소프트맥스 함수: 로짓을 입력으로 받아 각 클래스에 대한 확률을 계산하는 함수



입력층 뉴런의 수는 입력데이터 변수의 개수와 동일



출력층 뉴런의 수는 분류 분석의 경우 분류 레이블의 수이며, 회귀분석의 경우 1개

X

$$\hat{y} \approx y$$

차이(Loss, Error)

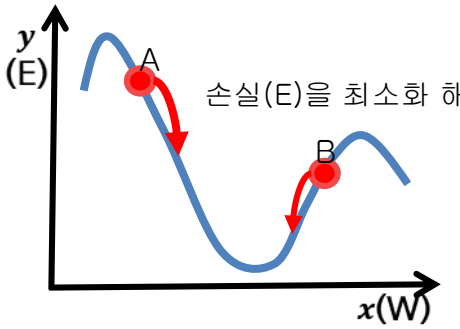
- MAE
- MSE

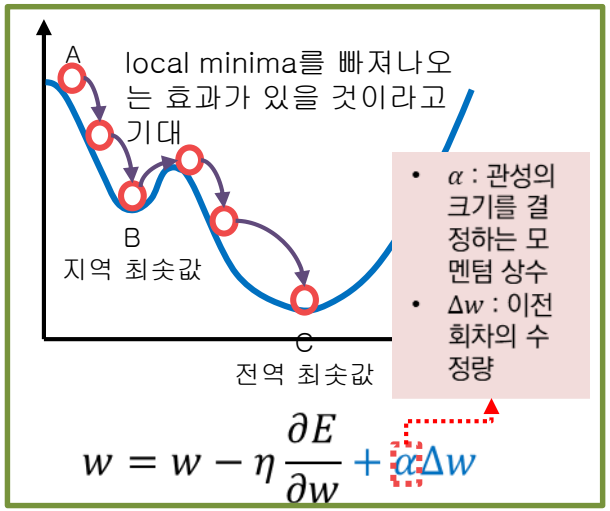
회귀

- Crossentropy

분류

손실함수





가중치 업데이트

Gradient Descent

$$w = w - \eta * \frac{\partial E}{\partial w}$$

- weight
- 네트워크의 파라미터

- Eta(learning rate)
- 한번에 얼마나 학습할지

- w에 대한 E의 편미분
- gradient
- 어떤 방향으로 학습할지

- 편미분(偏微分, partial derivative)을 의미
- 기호는 델타의 변형
- 편-디(partial dee), 파셜디라고 부름

$$h = h + \left(\frac{\partial E}{\partial w}\right)^2$$

$$w = w - \eta \frac{1}{\sqrt{h}} \frac{\partial E}{\partial w}$$

연구자가 고려해야 할 사항

- 은닉층의 수
- 은닉층별 뉴런의 수
- 활성화 함수
- 가중치 초깃값
- 드롭아웃 비율
- 배치정규화
- 손실함수
- 옵티마이저, 학습률
- 학습 횟수(epochs)
- 배치 크기

ChatGPT-3의 파라미터 개수는? 175B
ChatGPT-4의 파라미터 개수는? 1.7T

2장. 텐서플로우(케라스)를 이용한 딥러닝 구현

인공신경망 딥러닝 알고리즘 구현과 실시간 객체 탐지

1절. 텐서플로우 케라스

2장. 텐서플로우 케라스를 이용한 딥러닝 구현

케라스와 파이토치 비교

케라스와 파이토치 비교

Keras

- 높은 수준의 API를 제공함
- 코드가 직관적이고 사용하기 쉬움
- TensorFlow의 고수준 API로 통합되었음
- TensorFlow 생태계를 잘 활용할 수 있음

PyTorch

- 낮은 수준의 API를 제공
- 더 유연하고 커스터마이징하기 좋음
- 연구 커뮤니티에서 강력한 지지를 받고 있음
- 연구 결과를 빠르게 구현할 수 있는 라이브러리나 도구가 많음

기능	케라스	파이토치
모델 정의	Sequential 또는 Functional API 사용	nn.Module 클래스 상속
레이어 추가	model.add 또는 리스트 내 정의	__init__ 메서드 내에 레이어 정의
순전파	자동으로 처리	forward 메서드 내에서 직접 정의
모델 컴파일	model.compile 사용	없음, 직접 손으로 해야 함
모델 훈련	model.fit 사용	훈련 루프 직접 작성
손실 함수	loss 인자로 지정	nn.CrossEntropyLoss 등 손실 함수를 직접 정의
최적화 알고리즘	optimizer 인자로 지정	optim.Adam 등 옵티마이저를 직접 정의

텐서플로우 케라스

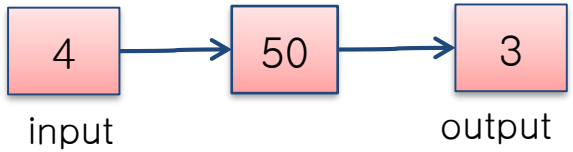
- 텐서플로우의 케라스 API는 딥러닝 모델을 구축하고 훈련하기 위한 고수준 API로, 다양한 신경망 구조를 쉽게 구축할 수 있도록 설계되었음
- 텐서플로우 2.x이후부터 공식적으로 케라스를 포함함

- tf.keras.Sequential: 순차 모델은 여러 층을 연이어 쌓아서 신경망을 구축하는 데 사용됨. 각 층은 이전 층의 출력을 입력으로 받음
- tf.keras.Model: 사용자 정의 모델을 만들 때 이 클래스를 상속하여 모델을 정의할 수 있습니다. 이를 사용하면 복잡한 모델 구조를 구현할 수 있음.
- tf.keras.layers: 이 모듈은 다양한 종류의 층을 포함하고 있으며, 이를 사용하여 모델의 구조를 정의함. 예를 들어, Dense 층은 완전 연결된 층을 생성하고, Conv2D 층은 합성곱 신경망을 생성함
- tf.keras.losses: 손실 함수를 정의하는 데 사용. 모델의 훈련 중에 사용되며, 훈련 중에 최소화하고자 하는 목표 함수를 정의함. 예를 들어, categorical_crossentropy는 다중 클래스 분류 문제의 크로스 엔트로피 손실 함수임
- tf.keras.optimizers: 최적화 알고리즘을 선택하고 모델을 훈련하기 위해 사용. 예를 들어, Adam, SGD, RMSprop와 같은 최적화 알고리즘을 제공함.
- tf.keras.metrics: 모델의 성능을 평가하기 위한 메트릭을 정의하는 데 사용. 예를 들어, 정확도(accuracy)나 정밀도(precision)를 계산할 수 있음
- tf.keras.callbacks: 모델 훈련 중에 호출되는 콜백 함수를 정의하는 데 사용. 예를 들어, 학습률을 동적으로 조정하거나 모델 저장을 자동화하는 데 유용함
- tf.keras.utils: 데이터 전처리 및 유틸리티 함수를 제공하는 모듈. 데이터 로드, 정규화, 원-핫 인코딩 등의 작업을 처리할 때 유용

Sequential 클래스 이용

예

```
1. from tensorflow.keras import Sequential
2. from tensorflow.keras.layers import Input, Dense
3.
4. model = Sequential()
5. model.add(Input(shape=4))
6. model.add(Dense(50, activation='relu'))
7. model.add(Dense(3, activation='softmax'))
8. model.summary()
```




```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                250

 dense_1 (Dense)             (None, 3)                 153

=================================================================
Total params: 403 (1.57 KB)
Trainable params: 403 (1.57 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

장점

- Sequential API로 모델을 설계하는 것은 직관적이고 편리함

단점

- 단순히 층을 쌓는 것만으로는 모든 인공신경망을 구현할 수 없음. 즉, 복잡한 인공 신경망을 구현할 수 없음.

함수형 API 이용

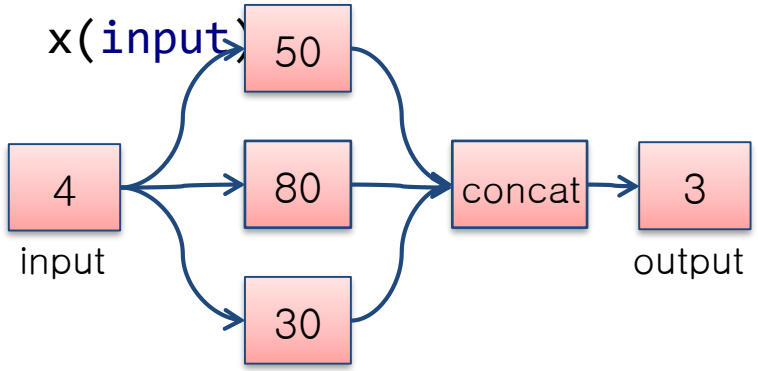
함수형 API를 이용하면 다중 입력 및 다중 출력 모델 생성 가능

```
1.  from tensorflow.keras import Model
2.  from tensorflow.keras.layers import Input, Dense
3.  from tensorflow.keras.layers import concatenate, Activation
4.
5.  input = Input(shape=(4,))
6.  dense1 = Dense(50, activation='relu')(input)
7.  dense2 = Dense(80, activation='relu')(input)
8.  dense3 = Dense(30, activation='relu')(input)
9.  x = concatenate([dense1, dense2, dense3])
10. output = Dense(3, activation='softmax')(x)
11. model = Model(inputs=input, outputs=output)
12. model.summary()
```

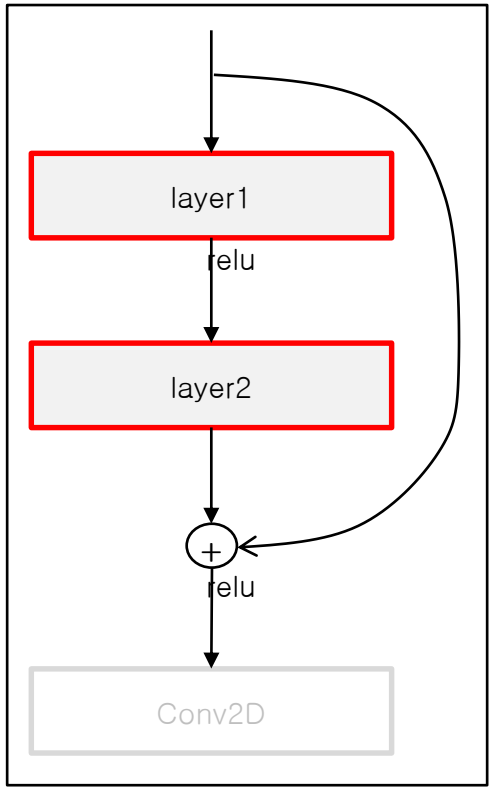
```
x = Dense(64)(input)
```

위의 코드는 아래의 코드와 같음

```
x = Dense(64)
```



Residual block 구현



4장. 합성곱 신경망

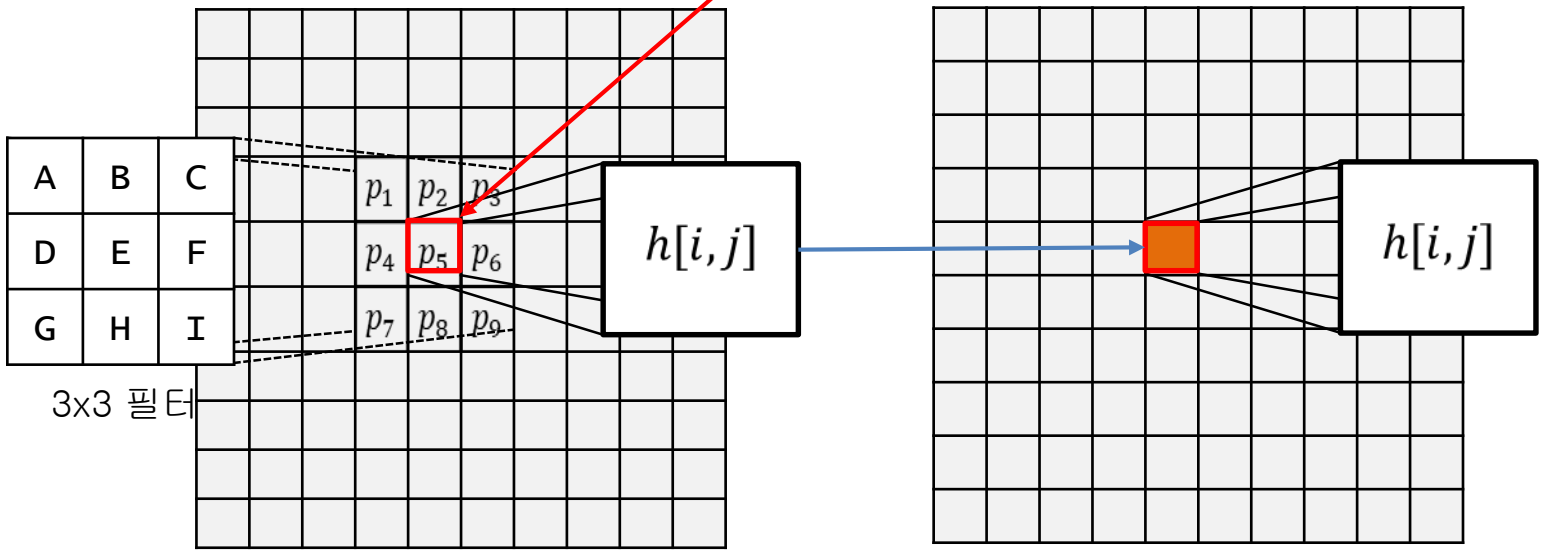
인공신경망 딥러닝 알고리즘 구현과 실시간 객체 탐지

1절. 합성곱 신경망

이미지 합성곱

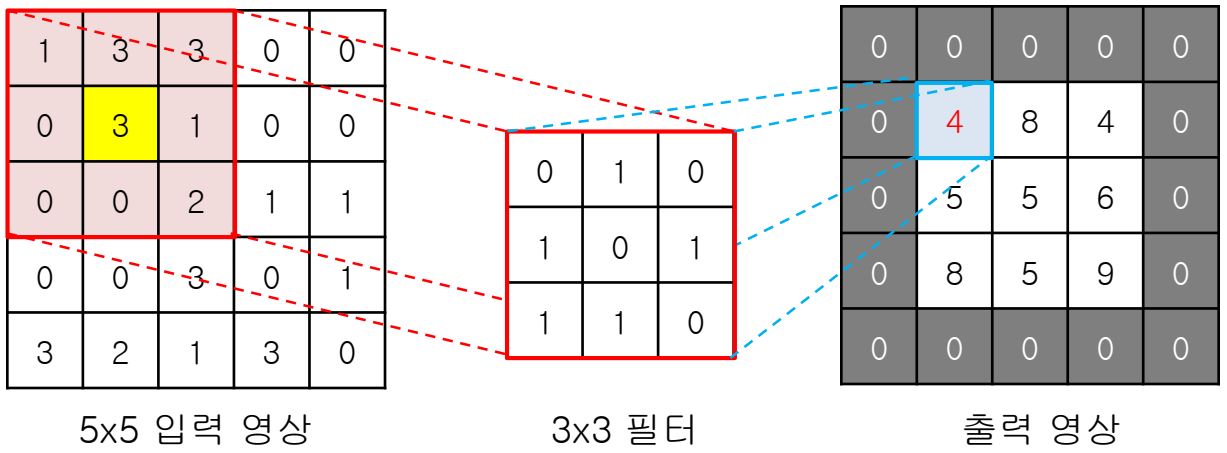
이미지는 2차원 이므로 적용하는 필터도 2차원

필터를 적용할 수 있는 영역에 대해서 현재 주목화소와 주위 화소의 배열을 각 요소끼리 모두 곱한 후 모든 항목을 더함



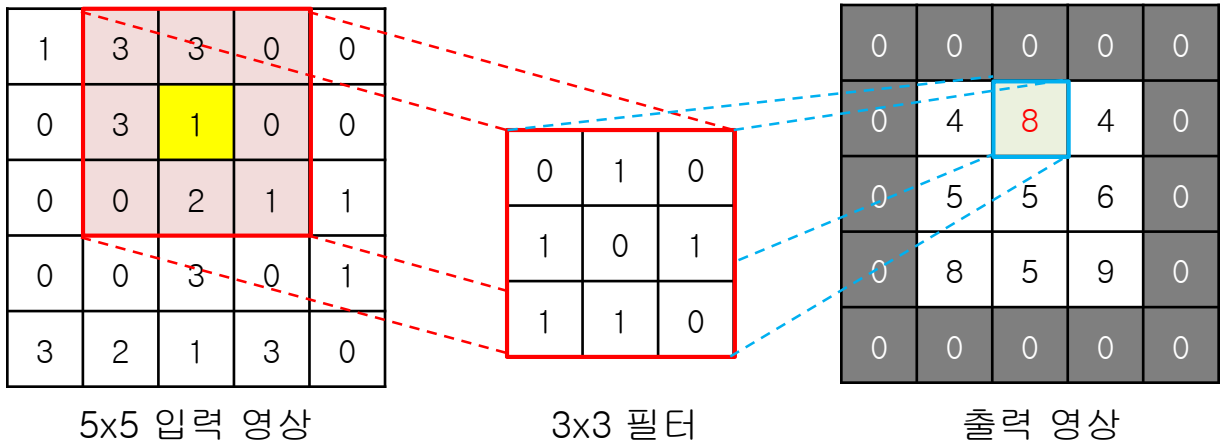
$$h[i,j] = A \times p_1 + B \times p_2 + C \times p_3 + D \times p_4 + E \times p_5 + F \times p_6 + G \times p_7 + H \times p_8 + I \times p_9$$

필터 적용



Zero padding : 5x5이미지의 바깥 화소는 필터를 적용할 수 없으므로 0으로 채움

img[1,1] = (1x0) + (3x1) + (3x0) + (0x1) + (3x0) + (1x1) + (0x1) + (0x1) + (2x0) = 4



img[1,2] = (3x0) + (3x1) + (0x0) + (3x1) + (1x0) + (0x1) + (0x1) + (2x1) + (1x0) = 8

이미지 필터 적용 예

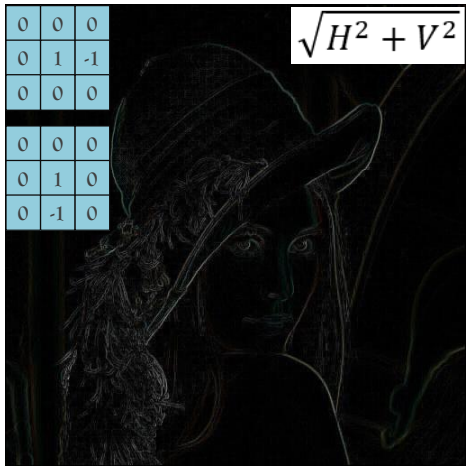
필터 커널에 따라 원본 이미지로부터 특성이 강조된 이미지를 얻을 수 있음

원 영상

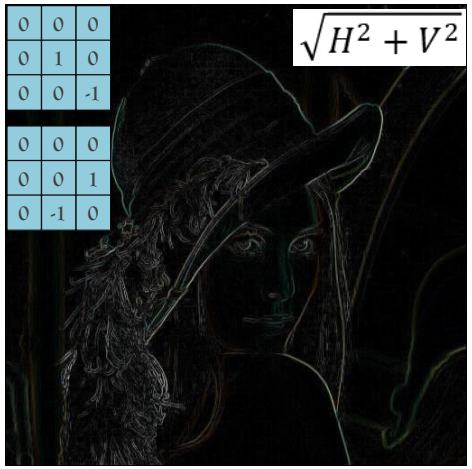
Sharpen 필터

Box blur

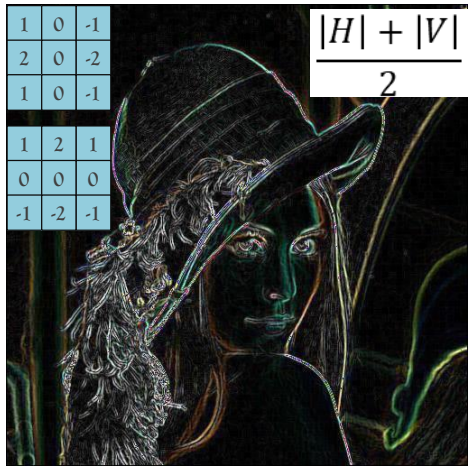
Gaussian blur



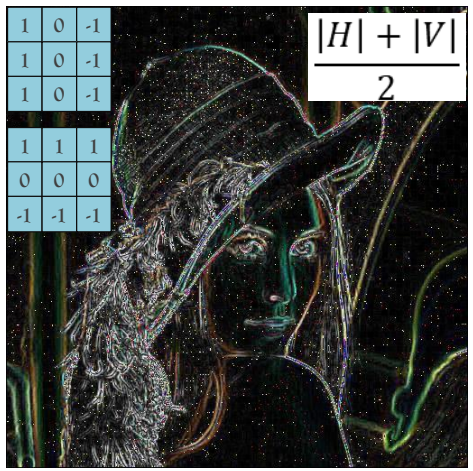
차분 필터



Roverts 필터



Sobel 필터



Prewitt 필터

필터 적용 전/후

1절. 합성곱 신경망 > 1.2. 이미지 합성곱

0	0	0
0	1	-1
0	0	0

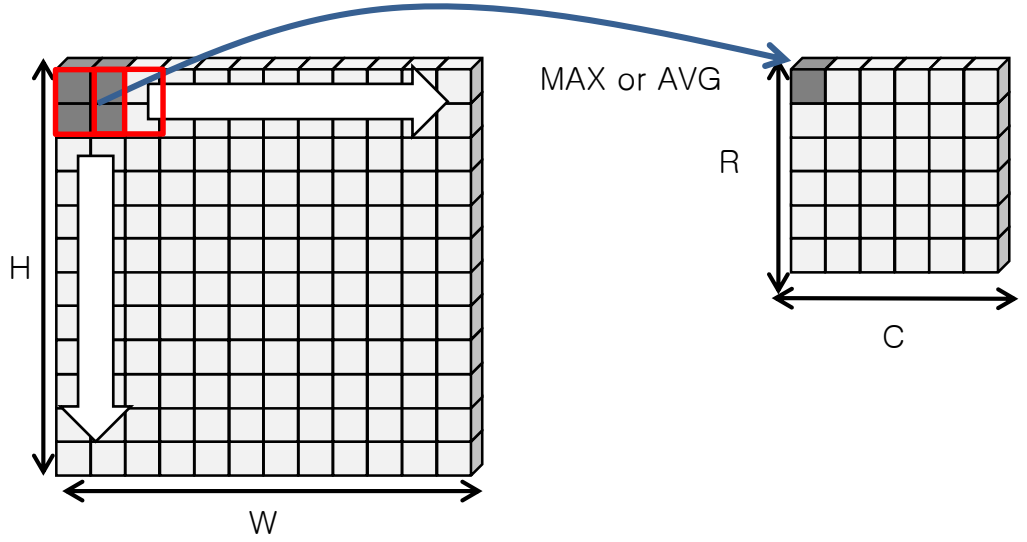
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	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	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	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	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	253	253	187	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	249	253	249	64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	298	81	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	136	253	253	253	212	135	132	16	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	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
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	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	3	15	0	0	118	10	39	149	140	89	8	120	127	0	0	0	0
0	0	0	0	0	0	0	30	6	58	60	16	83	0	0	0	0	28	47	71	9	47	131	64	0	0	0	0
0	0	0	0	0	0	49	189	15	0	0	0	0	0	0	0	2	158	11	0	26	17	39	0	0	0	0	0
0	0	0	0	0	0	18	201	34	0	0	0	0	55	16	65	6	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	80	86	49	46	0	48	194	11	43	111	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	14	13	153	99	163	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	139	14	63	188	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	11	179	63	183	70	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	35	206	16	65	52	107	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	81	159	13	0	34	94	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	45	141	67	0	103	123	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	77	160	0	66	187	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	4	4	185	64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	46	84	53	70	0	46	205	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	39	109	81	24	0	0	3	68	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	24	90	107	32	0	0	0	52	123	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	23	43	147	40	0	0	0	55	217	79	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	18	153	48	34	0	0	0	58	115	71	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	55	117	54	27	0	0	0	9	111	122	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	136	117	0	0	41	77	3	116	16	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	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

현재 화소와 현재 화소 주위 화소들을 이용해 필터링 처리 해야 하므로 이미지의 [1,1] 화소부터 [27,27]화소 까지만 필터링 처리 할 수 있다. [0, 0]화소는 x좌표, y좌표가 음수 값이 없다.

풀링

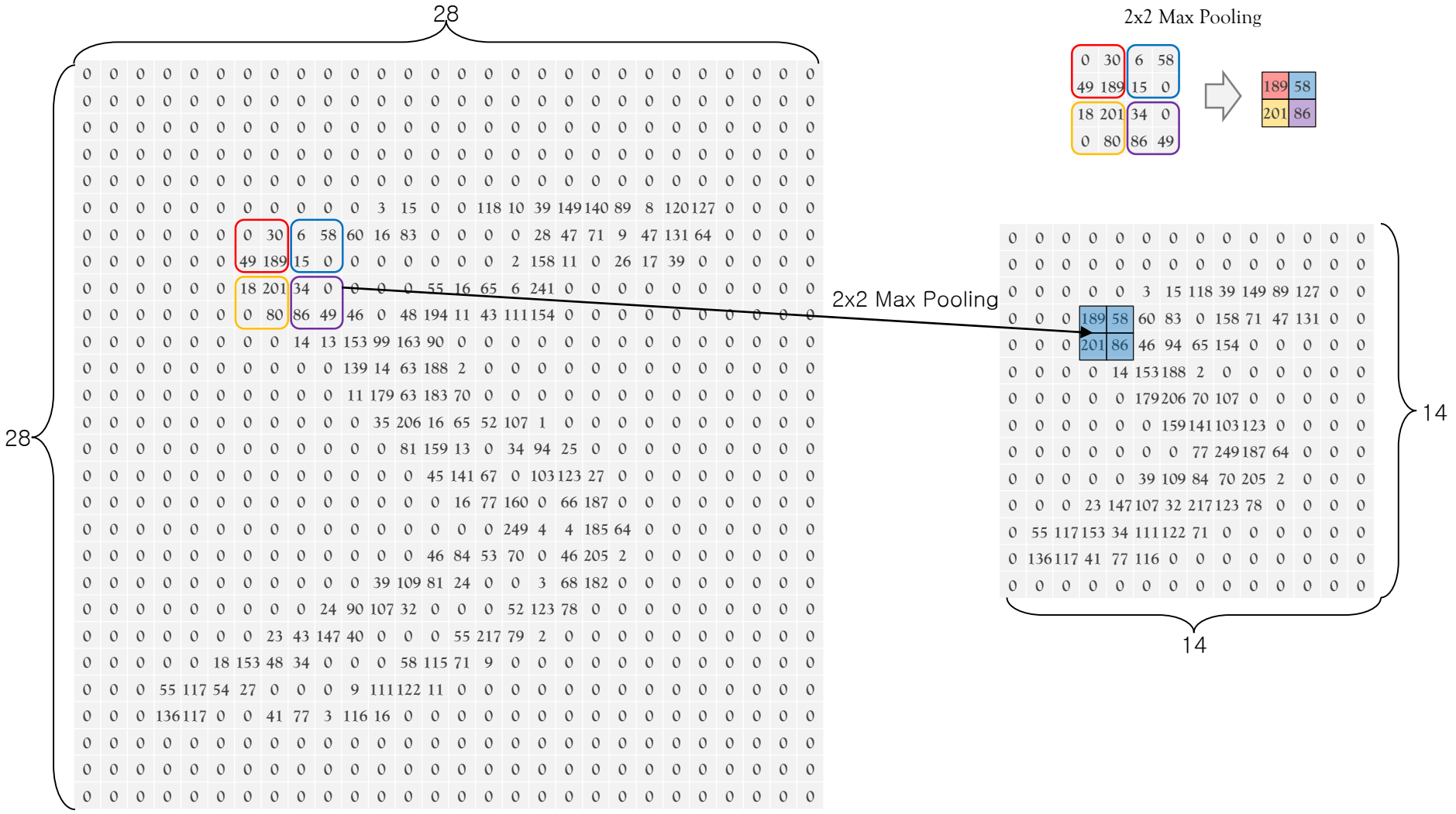
입력 이미지를 차원 축소해서 크기를 줄이는 것



풀링(Pooling) 방법

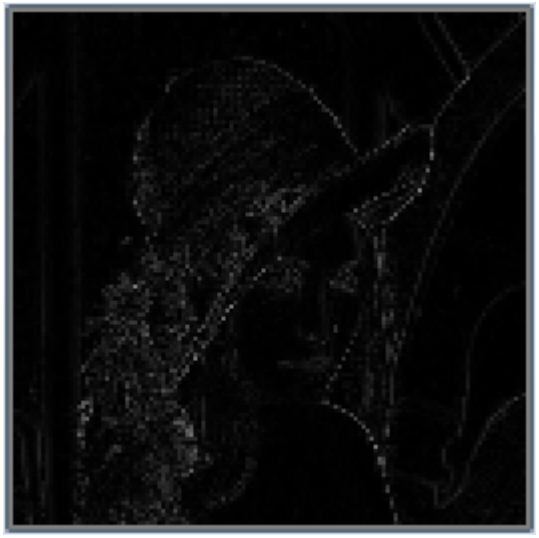
- Max Pooling : 대상 화소 중에서 가장 큰 화소 값을 선택
- Mean Pooling(Average Pooling) : 대상 화소들의 평균값을 선택
- Min Pooling : 대상 화소 중에서 가장 작은 값을 선택
- Overlapped Pooling : 풀링 사이즈보다 스트라이드가 작을 경우

Max Pooling



Resize vs. Max pooling

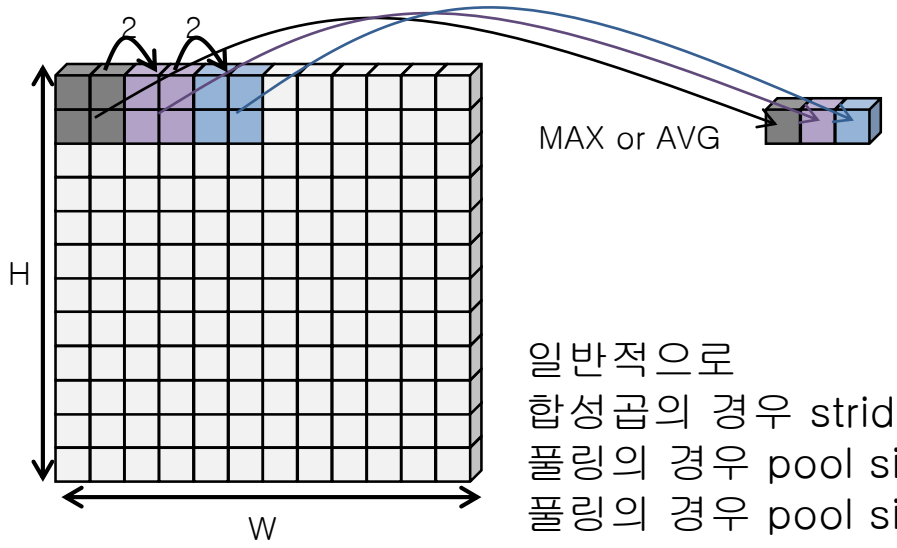
Resize 1/2



2x2 Max pooling

스트라이드

스트라이드 : 합성곱 또는 풀링 과정에서 필터가 이미지에 적용된 후 움직이는 크기

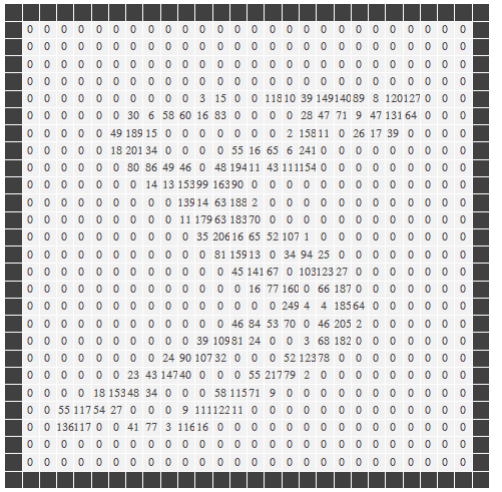


일반적으로
합성곱의 경우 strides=1
풀링의 경우 pool size와 같음
풀링의 경우 pool size 보다 작으면 overlapped pooling이라고 함

제로 패딩

0	0	0
0	1	-1
0	0	0

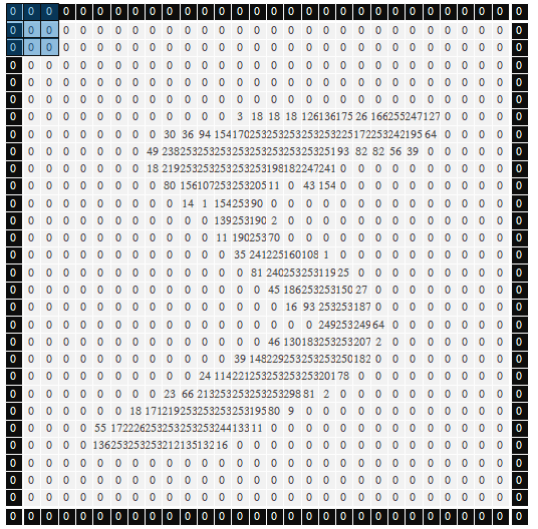
- 합성곱 연산에 의해 바깥 화소가 사라지는 것을 방지하기 위함
- 사라지는 픽셀 테두리 수는 floor(filter_size/2)
- 3x3필터를 합성곱 하면 바깐 1화소가 사라짐
- 제로패딩은 floor(filter_size/2)만큼 위/아래/왼쪽/오른쪽 화소가 더 있다고 가정하고 처리하는 것
- Conv2D() 클래스의 padding='same'으로 하면 제로패딩을 사용(기본값은 padding='valid')



필터가 3x3일 경우 바깥 1픽셀 정보가 사라짐

원 영상

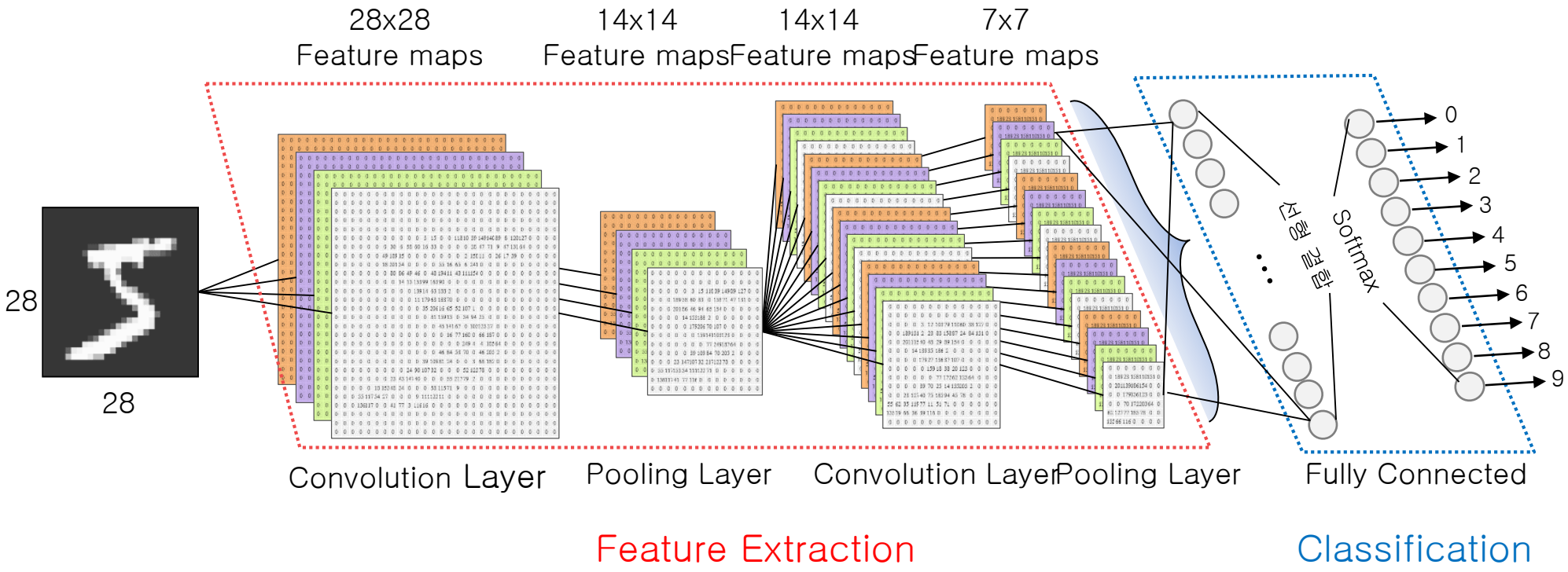
5x5 Median 필터 적용 후



바깥 1화소가 더 있다고 가정하고 처리

Stride는 한번에 움직이는 행렬의 크기

CNN 모델


28x28
Feature maps
14x14
Feature maps
14x14
Feature maps
7x7
Feature maps
28
28
Convolution Layer
Pooling Layer
Convolution Layer
Pooling Layer
Fully Connected
전결합
Softmax
0
1
2
3
4
5
6
7
8
9
Feature Extraction
Classification

2절. 손글씨 숫자 인식을 위한 CNN 구현

4장. 합성곱 신경망

손글씨 숫자 인식하기

- 우편번호 자동 분류기처럼 숫자를 분류할 수 있도록 만들기 위해 필요한 것
  - 딥러닝 구현 알고리즘
  - 손으로 쓴 숫자 데이터
- 알고리즘은?
  - 인공신경망 CNN 사용
- 데이터는?
  - 종이에 0~9까지 숫자를 쓴다. → 대략 1000개 이상은 써야 하지 않을까?
  - 숫자를 이미지로 저장한다.
  - 개별 숫자를 잘라내어 파일로 저장한다.
  - 전처리 한다.
    - 숫자의 크기를 일정한 크기로 자르고, 밝기 값 조정 등…
  - 레이블을 지정한다.

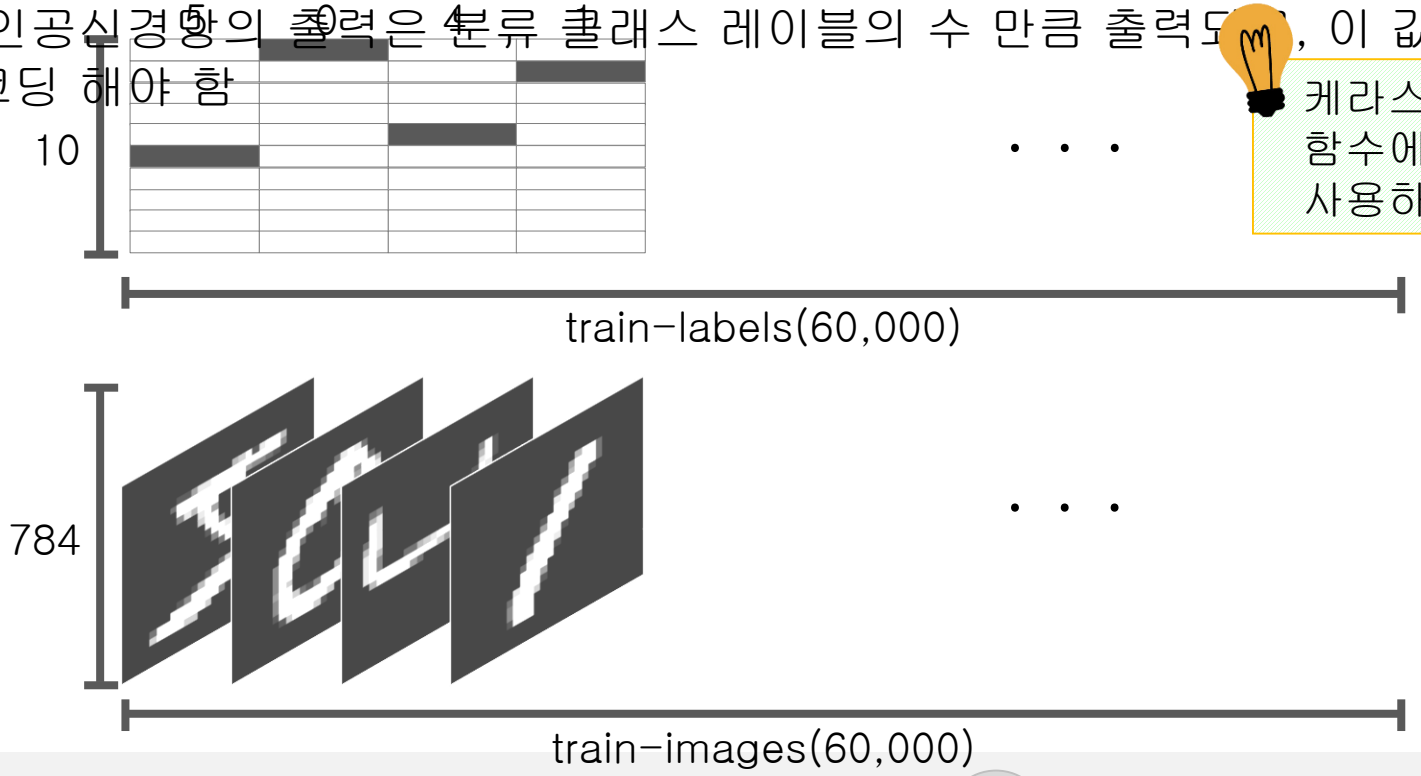
MNIST 손글씨 숫자 데이터

- MNIST(Modified National Institute of Standards and Technology)
- 미국의 NIST에서 이미지 처리 시스템을 위해 모은 0부터 9까지의 숫자 이미지 데이터셋
- 각 이미지는 28x28 크기, 이것을 펼치면 784 차원의 벡터

28 x 28 pixels image 이미지를 784개 화소(Pixel)의 숫자로 표현

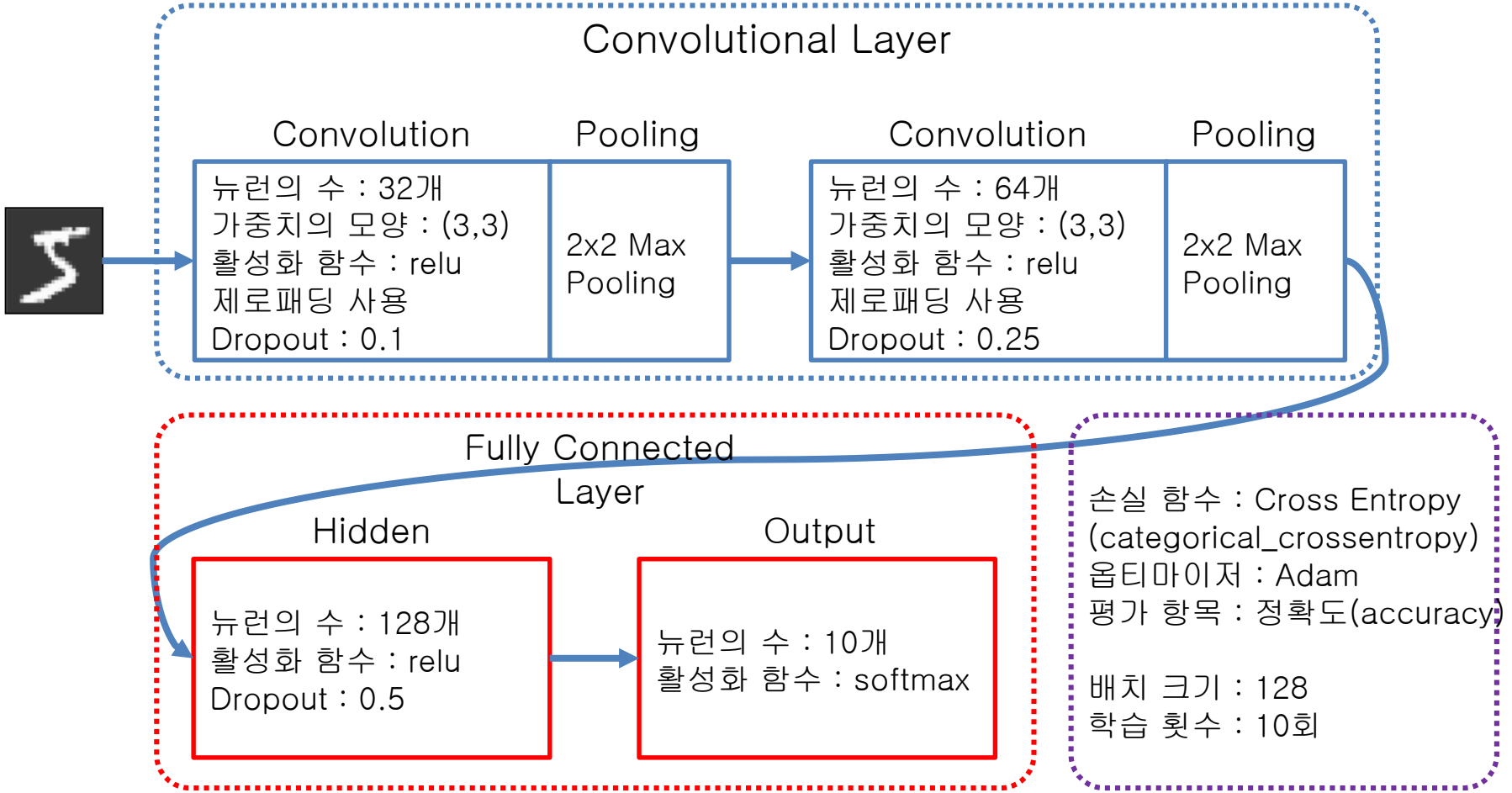
인공신경망에서 원-핫 인코딩

- 원-핫 인코딩(One-Hot Encoding)은 범주형 데이터(카테고리 데이터)를 수치 데이터로 변환하는 데 사용되는 기술
- 원-핫 인코딩의 핵심 아이디어는 각 카테고리 값을 이진 벡터로 표현하는 것
  - 각 카테고리는 해당하는 인덱스 위치의 원소만 1이고 나머지 원소는 모두 0으로 표현됨
  - 이렇게 하면 모든 카테고리가 수치 데이터로 변환되며, 각 카테고리 간에 상호간섭이 없게 됨
- 인공신경망은 멀티클래스 분류일 경우 출력층의 뉴런의 개수가 클래스 라벨의 수와 일치해야 하는데, 그러면 인공신경망의 출력은 분류 클래스 레이블의 수 만큼 출력되고, 이 값이 정답과 비교되려면 정답을 원-핫 인코딩 해야 함

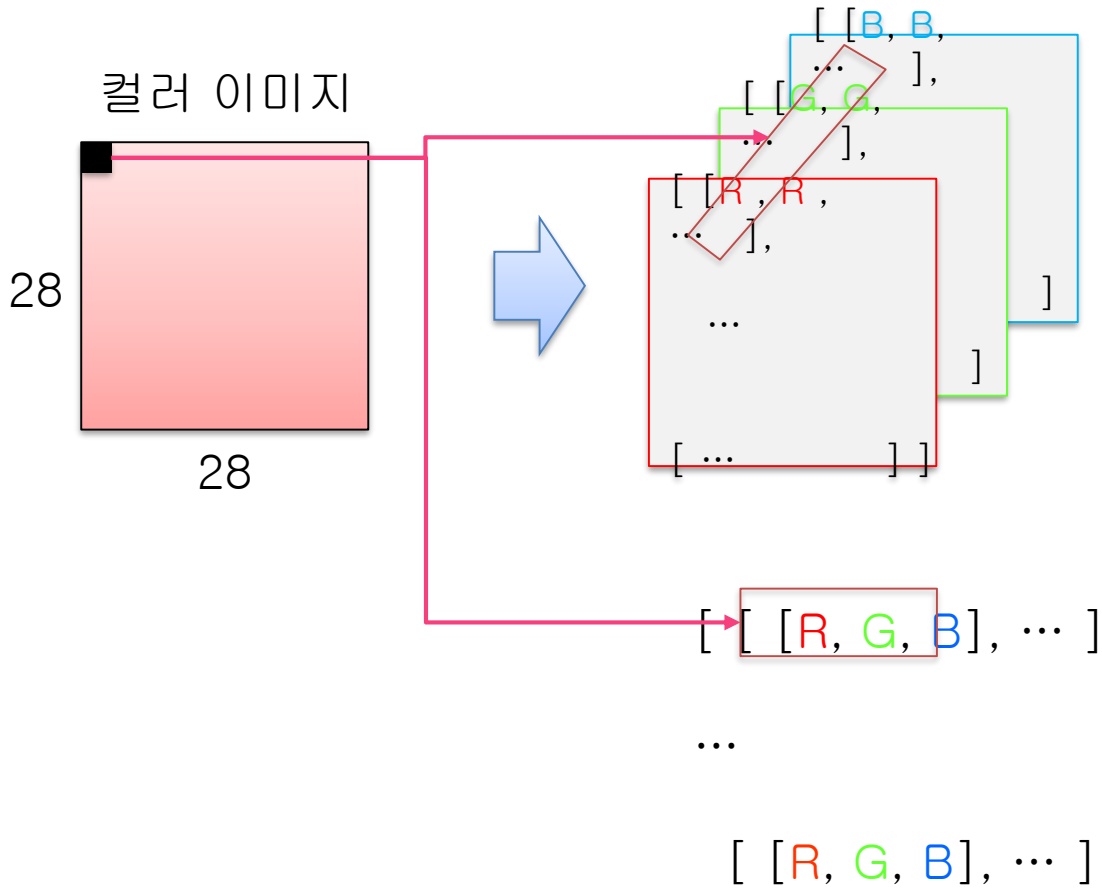


케라스에서는 멀티클래스 분류를 위한 손실 함수에 sparse_categorical_crossentropy를 사용하면 원-핫 인코딩 하지 않아도 됩니다.

손글씨 숫자 인식을 위한 케라스 CNN 모델



Channel First와 Channel Last



Channel First

(3, 28, 28)

- 텐서플로우는 Channel last, 파이토치는 Channel first 구조를 사용합니다.

Channel Last

(28, 28, 3)

CNN 코드 (1/2) – 모델 정의하기

모델 정의

```
1. from tensorflow.keras import Sequential, layers
2. model = Sequential([
3.     layers.Input(shape=(28,28,1)),
4.     layers.Conv2D(32, (3,3), activation='relu'),
5.     layers.MaxPooling2D(), # pool_size=(2,2)가
   기본값
6.     layers.Dropout(0.1),
7.     layers.Conv2D(64, 3, activation='relu'),
8.     layers.MaxPooling2D(),
9.     layers.Dropout(0.25),
10.    layers.Flatten(),
11.    layers.Dense(128, activation='relu'),
12.    layers.Dropout(0.5),
13.    layers.Dense(10, activation='softmax')
14.])
15.model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 32)        832

 max_pooling2d (MaxPooling2  (None, 12, 12, 32)        0
 D)

 dropout (Dropout)           (None, 12, 12, 32)        0

 conv2d_1 (Conv2D)           (None, 10, 10, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 dropout_1 (Dropout)         (None, 5, 5, 64)          0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 225546 (881.04 KB)
Trainable params: 225546 (881.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

CNN 코드 (2/2) – 데이터 불러오고 학습한 후 평가하기

훈련 정의

```
1. model.compile(loss='sparse_categorical_crossentropy',
2.                optimizer='adam', metrics=['accuracy'])
```

데이터 불러오기

```
1. from tensorflow.keras.datasets import mnist
2. (X_train, y_train), (X_test, y_test) = mnist.load_data()
3. X_train = X_train.reshape(-1, 28, 28, 1) / 255.0
4. X_test = X_test.reshape(-1, 28, 28, 1) / 255.0
```

데이터 학습

```
1. model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1)
```

모델 평가

```
1. loss, accuracy = model.evaluate(test_X, test_y, verbose=0)
2. print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

```
# 코랩에서 한글 그래프 적용하려면(세션 다시 시작)
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```