인공지능

딥러닝 중급_CNN

강성관 (silicon1@hanmail.net)

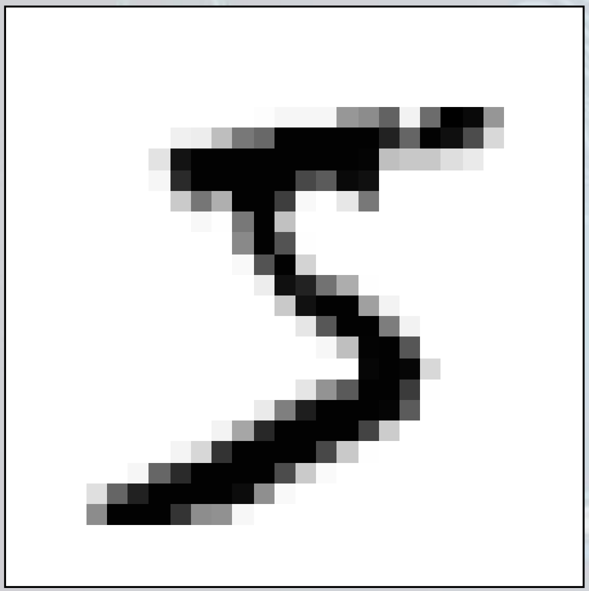
CNN

이미지 인식

- MNIST 데이터셋
  - 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손글씨를 이용해 만든 데이터로 구성
  - 70,000개의 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터셋
  - 뉴욕대 Yann LeCun 교수가 제공
  - 0~9까지의 숫자 이미지로 구성
  - 28 x 28 크기의 회색조 이미지로 총 784개의 픽셀로 구성
  - 시험 이미지 : 60,000장
  - 훈련 이미지 : 10,000장

이미지 인식

- 각 숫자 이미지는 28x28 크기의 총 784개의 픽셀로 구성
- 픽셀을 밝기 정도에 따라 0(흰색)~255(검정색)까지의 숫자로 구성되어 있음.



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	252	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	198	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

Sequential model API 살펴보기

- **compile()** : 훈련을 위한 모델을 구성

compile(optimizer, loss=None, metrics=None, loss_weights=None, sample_weight_mode=None, weighted_metrics=None, target_tensors=None)

- optimizer : 사용할 경사하강법 (SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam)
- loss : 사용할 손실함수 (mean_squared_error, categorical_crossentropy, binary_crossentropy 등)
- metrics : 평가항목 (일반적으로 accuracy)
- loss_weights : 손실 가중치
- sample_weight_mode : 시간에 따른 손실 가중치 설정
- weighted_metrics : 가중치 목록
- target_tensors : 사용할 목표 텐서

Sequential model API 살펴보기

- **fit()** : 모델을 훈련

```
fit(x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None,
validation_split=0.0, validation_data=None, shuffle=True, class_weight=None,
sample_weight=None, initial_epoch=0, steps_per_epoch=None,
validation_steps=None)
```

- x, y : 훈련데이터와 라벨
- batch_size : 한 번에 훈련할 데이터 크기
- epochs : 훈련 횟수
- vervose : 훈련과정 출력여부 (0: 미출력, 1: 출력(진행 바), 2: 출력(에포트 당 한 줄)
- callbacks : 학습 과정 데이터 관찰에 사용할 콜백함수 (ModelCheckPoint, EarlyStopping 등)
- validation_split : 학습 데이터의 비율
- validation_data : 사용할 학습 데이터, validation_split 설정이 무시됨
- shuffle : 각 에포크마다 데이터를 섞을 것인지 여부
- class_weight : 매핑된 클래스의 손실함수 가중치
- sample_weight : 샘플의 손실함수 가중치
- initial_epoch : 이전 훈련을 다시 실행하는 지점 설정
- steps_per_epoch : 에포크 당 반복 수 (50개 훈련 데이터가 있고 배치크기가 10이면 5스텝)
- validation_steps : steps_per_epoch 설정 시 사용, 한 에포크 종료 시마다 사용되는 검증 스텝 수 (20개의 테스트 데이터가 있고 배치크기가 5이면 5 스텝)

Sequential model API 살펴보기

- **evaluate()** : 모델 평가를 수행하고 손실 값과 metrics 값을 반환

```
evaluate(x=None, y=None, batch_size=None, verbose=1, sample_weight=None, steps=None)
```

- x, y : 테스트 데이터와 라벨
- batch_size : 한번에 평가할 샘플의 수
- verbose : 평가 과정 출력 여부 (0: 미출력, 1: 출력(진행 바))
- sample_weight : 샘플 데이터의 가중치
- steps : 평가 완료까지의 반복 수

Sequential model API 살펴보기

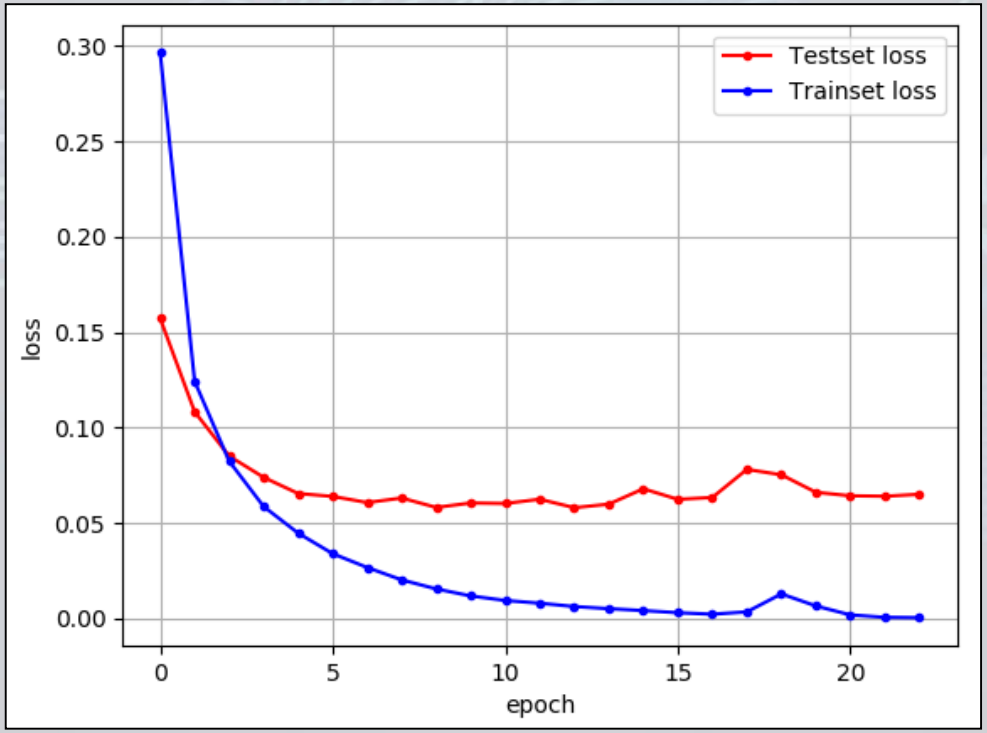
- **predict()** : 입력 데이터와 모델을 이용하여 예측 수행하고 예측 결과 반환

```
predict(x, batch_size=None, verbose=0, steps=None)
```

- x : 입력 데이터
- batch_size : 배치 크기
- verbose : 출력모드 (0: 미출력, 1: 출력)
- steps : 예측을 하기 까지의 총 반복 수

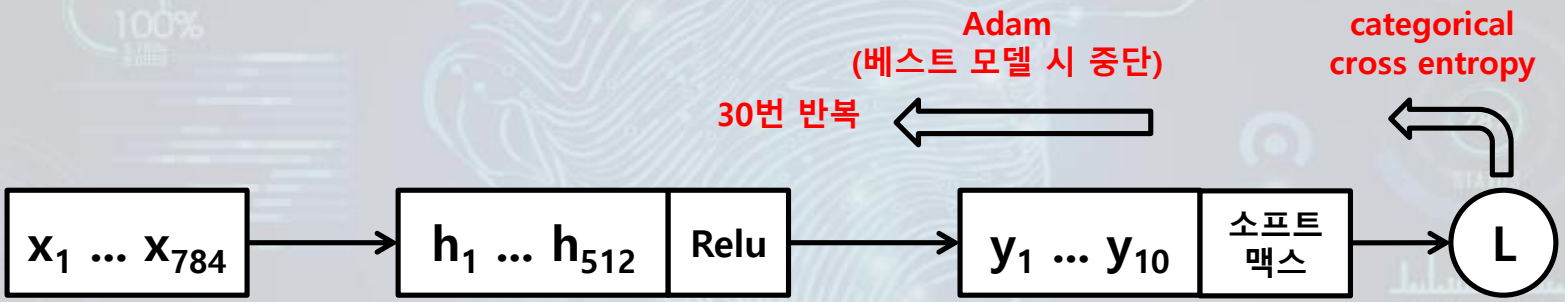
이미지 인식

- 학습 셋에 대한 오차는 계속 줄어들고 있으며 테스트 셋이 과적합이 일어나기 전에 학습을 종료
- 베스트 모델은 12번째 에포크일 때였으며 훈련오차는 0.009, 정확도는 99.86%, 테스트오차는 0.06, 정확도는 98.22%였음.



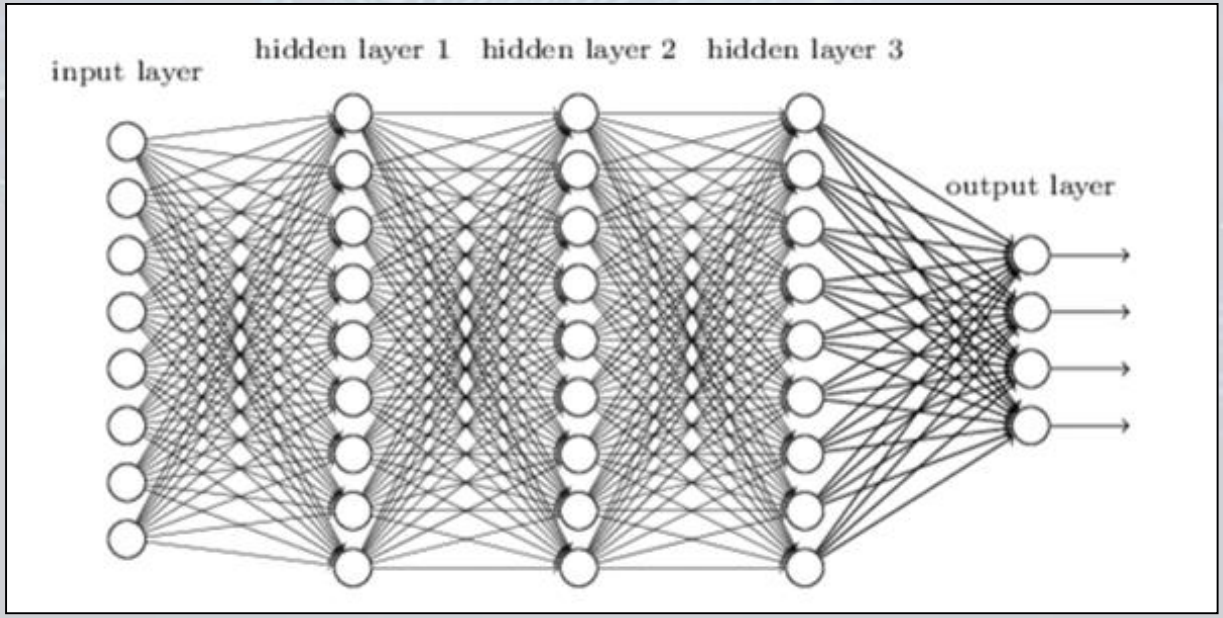
이미지 인식

- 이전의 딥러닝 프레임은 하나의 은닉층을 둔 단순한 모델이었음.
- 딥러닝은 기본 프레임을 바탕으로 프로젝트에 맞춰서 어떤 옵션을 더하고 어떤 층을 추가하느냐에 따라 성능이 좋아질 수 있음.



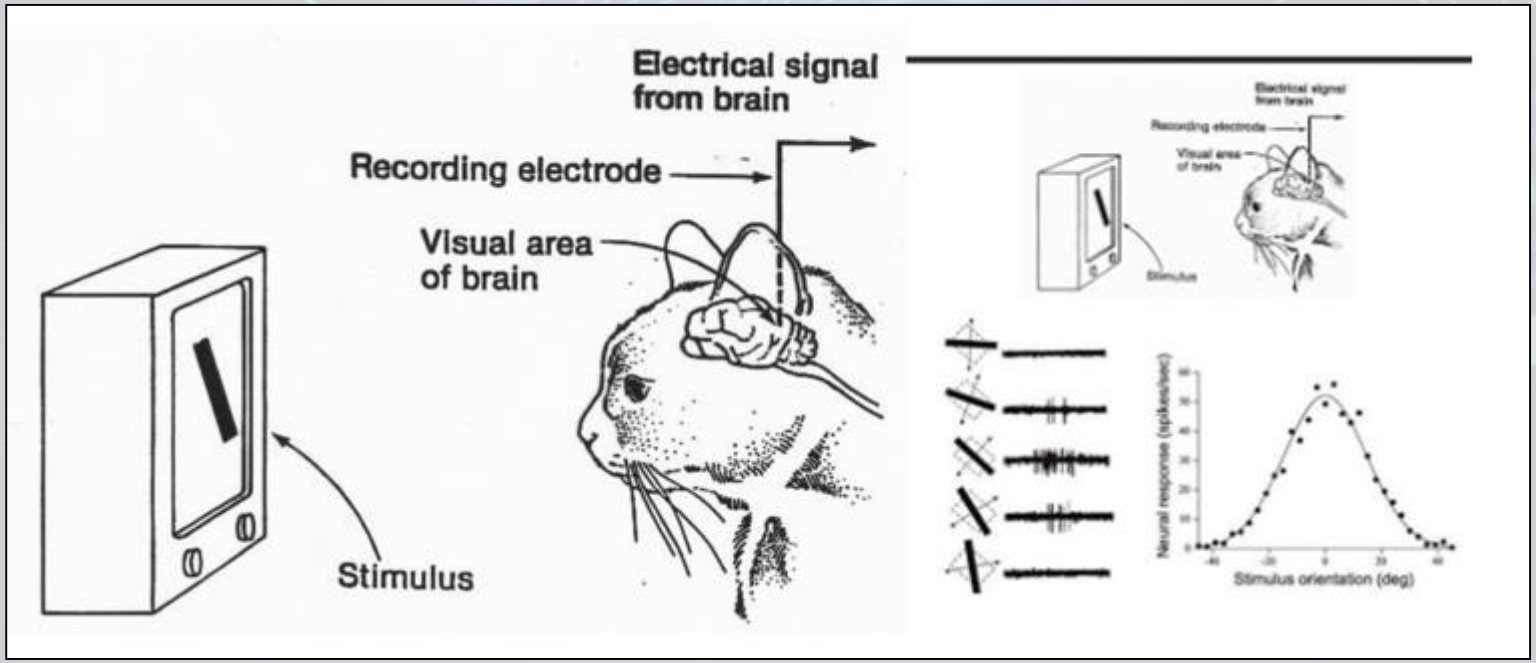
DNN (Deep Neural Network)

- Hidden Layer를 늘려서 컴퓨터 스스로가 레이블을 만들고 공간을 왜곡하고 데이터를 구분짓는 과정을 반복하여 최적의 해를 도출 → 많은 데이터와 반복학습이 필요 → Pre-training와 Back-propagation을 통해 최적의 학습 방법으로 사용
- 대표적인 알고리즘 : CNN, RNN, LSTM, GRU 등



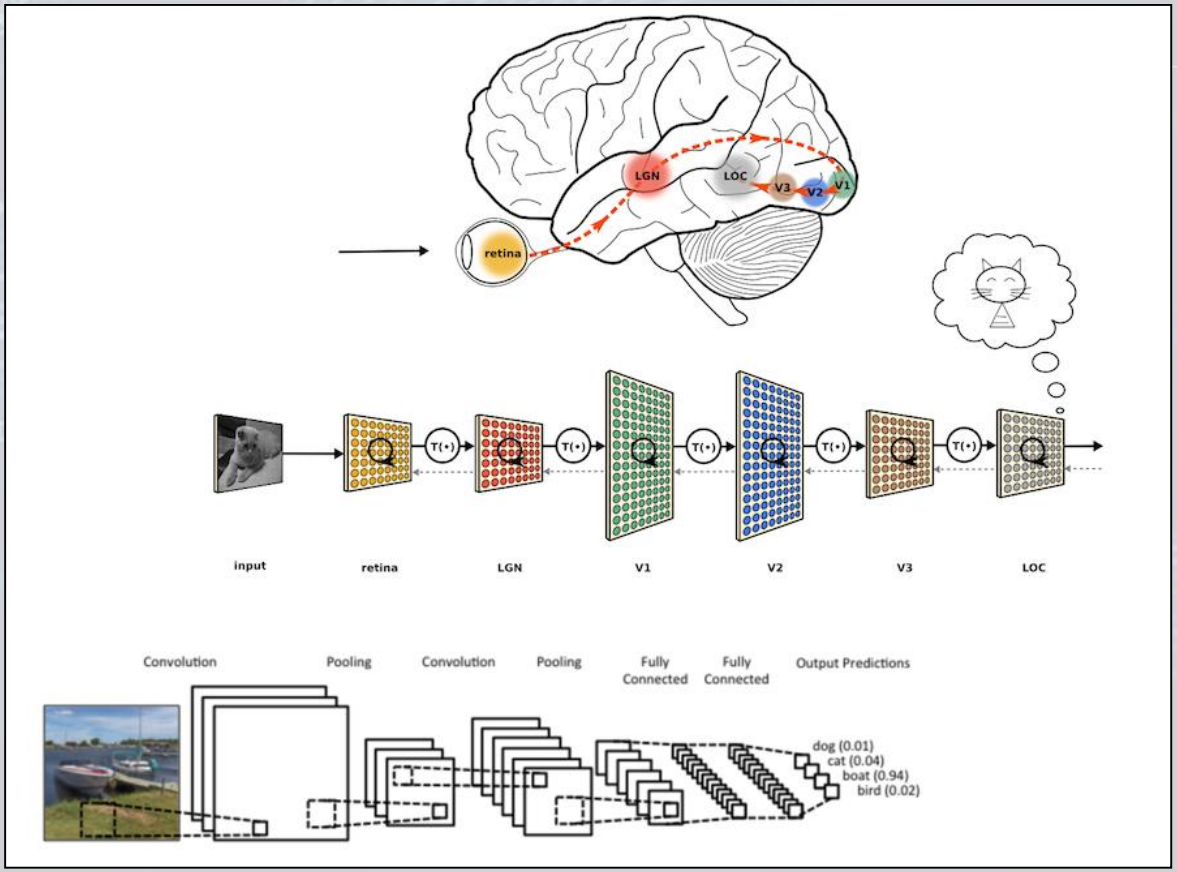
CNN (Convolution Neural Network)

- 1998년 Yann Lecun 교수에 의해 1950년 대 수행했던 고양이의 뇌파 실험에 영감을 얻어 이미지 인식을 획기적으로 개선할 수 있는 CNN 제안

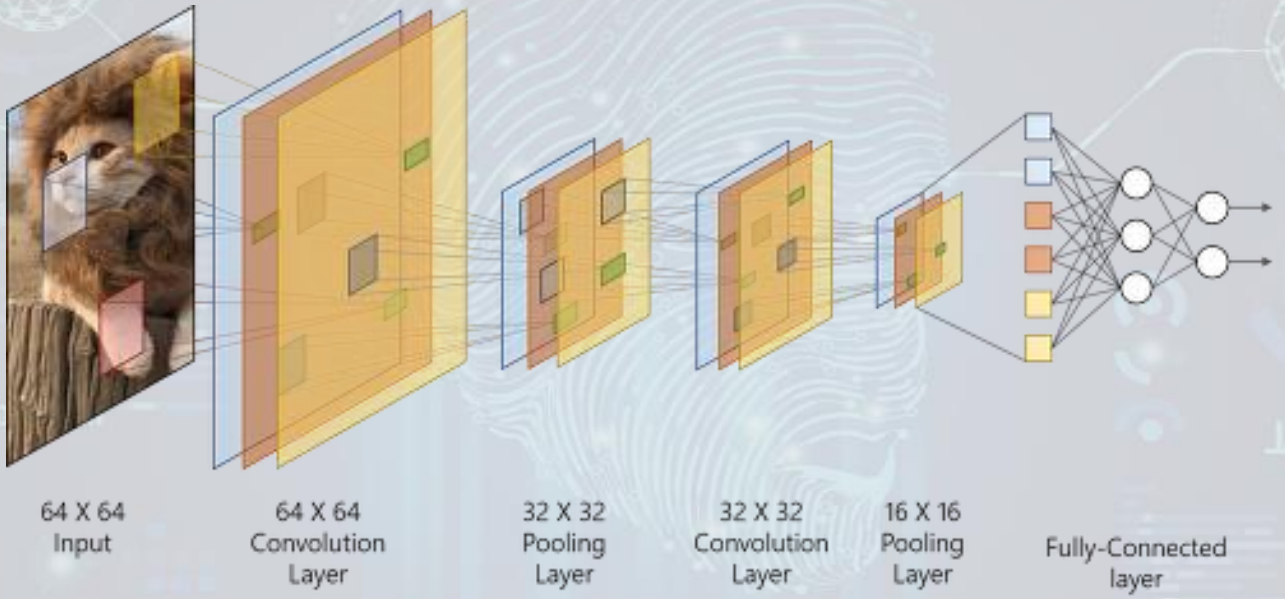


CNN (Convolution Neural Network)

- 고양이의 눈으로 보는 사물의 형태에 따라 뇌의 특정영역 (뉴런)만 활성화된다는 결과를 기반으로 제안

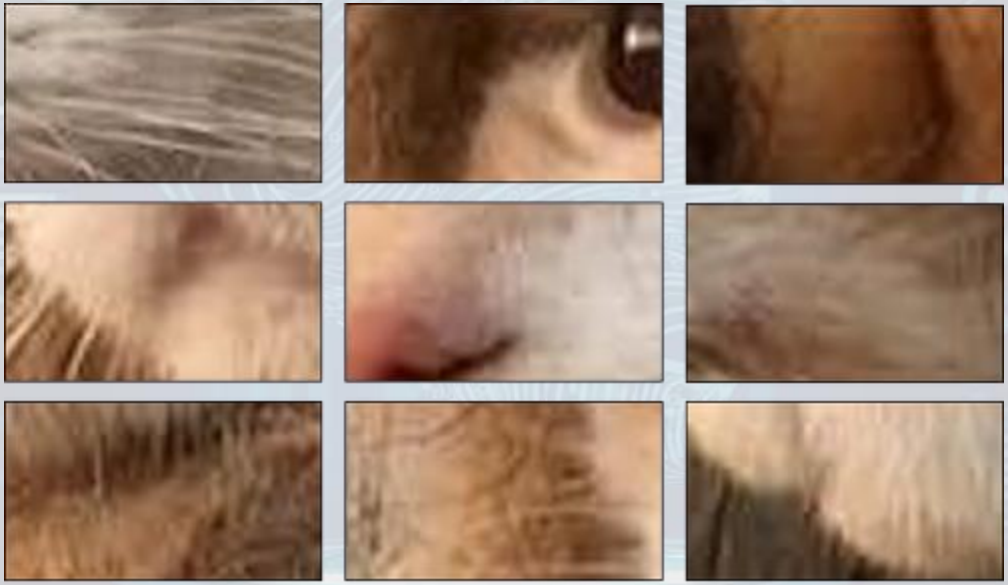


CNN의 구조


64 X 64
Input
64 X 64
Convolution
Layer
32 X 32
Pooling
Layer
32 X 32
Convolution
Layer
16 X 16
Pooling
Layer
Fully-Connected
layer

CNN의 구조

이미지를 가까운 거리에서 본다면



1단계 : 가로, 동그라미, 세모, 부드러움

CNN의 구조

조금 더 떨어져서 보면

2단계 :눈, 코, 귀, 발

CNN의 구조

더 떨어져서 보면

3단계 :고양이!

CNN의 구조

조각 이미지를 보고, 패턴을 익히고
점점 멀리서 조합을 하게 한다면

컴퓨터도 이미지를 인식할 수 있겠죠 ?

CNN의 구조

CNN (Convolution Neural Network)

- CNN (합성곱)은 입력된 이미지에서 다시 한번 특징을 추출하기 위해 마스크 (필터, 윈도우, 커널로 표현하기도 함)를 도입하는 방법
- 영상에서의 특정 위치에 있는 픽셀들은 그 주변에 있는 일부 픽셀과 correlation 높고, 거리가 멀어지면 멀어질수록 그 영향은 감소
- 영상 전체 영역에 대해 서로 동일한 연관성(중요도)로 처리하는 대신에 **특정 범위에 한정에 처리한다면 훨씬 효과적일 것이라는 아이디어 제시**
- Convolution은 비전에서 주로 filter연산을 뜻하며 이미지의 특징(feature)를 찾기 위해 filtering을 수행

1	0	1	0
1	0	1	0
1	1	1	1
0	0	1	0

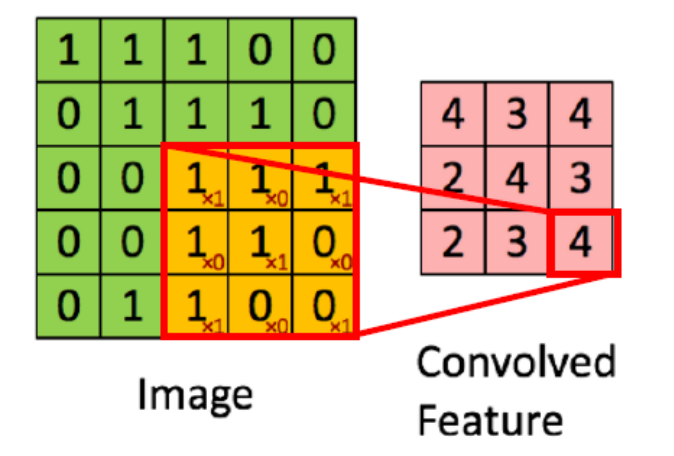
이미지

1	0
0	1

마스크

CNN (Convolution Neural Network)

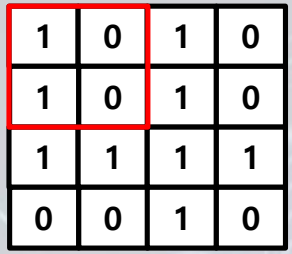
- 컨볼루션 과정
  - 데이터의 특징을 추출하는 과정으로 데이터에 각 성분의 인접 성분들을 조사해 특징을 파악
  - 파악한 특징을 한 장으로 도출시키는 과정 → Convolution layer
  - 이미지의 특정부분을 추상화 하여 특정 층으로 표현 → 압축과정으로 Parameter의 갯수를 효과적으로 줄여줌



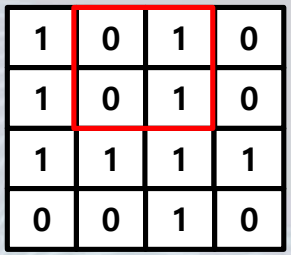
CNN (Convolution Neural Network)

- 컨볼루션 과정

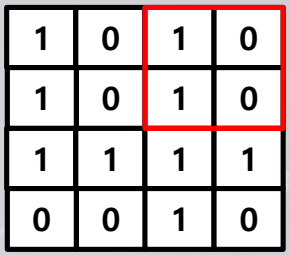
1	0
0	1



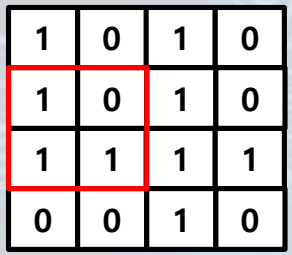
(1x1)+(0x0)+(0x1)+(1x0)=1



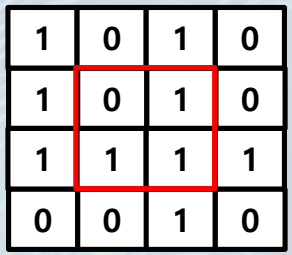
(0x1)+(1x0)+(0x0)+(1x1)=1



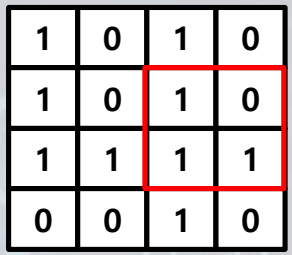
(1x1)+(1x0)+(0x0)+(0x1)=1



(1x1)+(0x0)+(0x1)+(1x1)=2

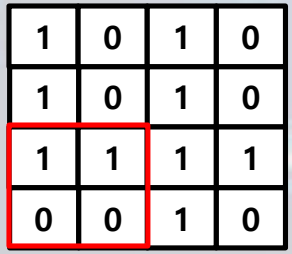


(1x0)+(0x1)+(0x1)+(1x1)=1

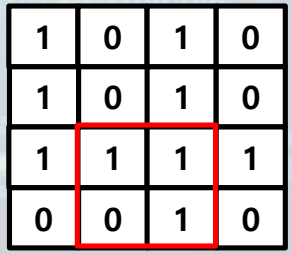


(1x1)+(0x0)+(0x1)+(1x1)=2

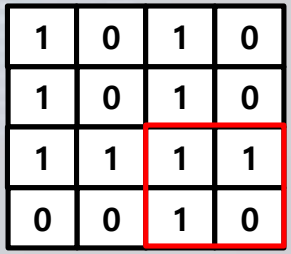
1	1	1
2	1	2
1	2	1



(1x1)+(0x1)+(0x0)+(1x0)=1



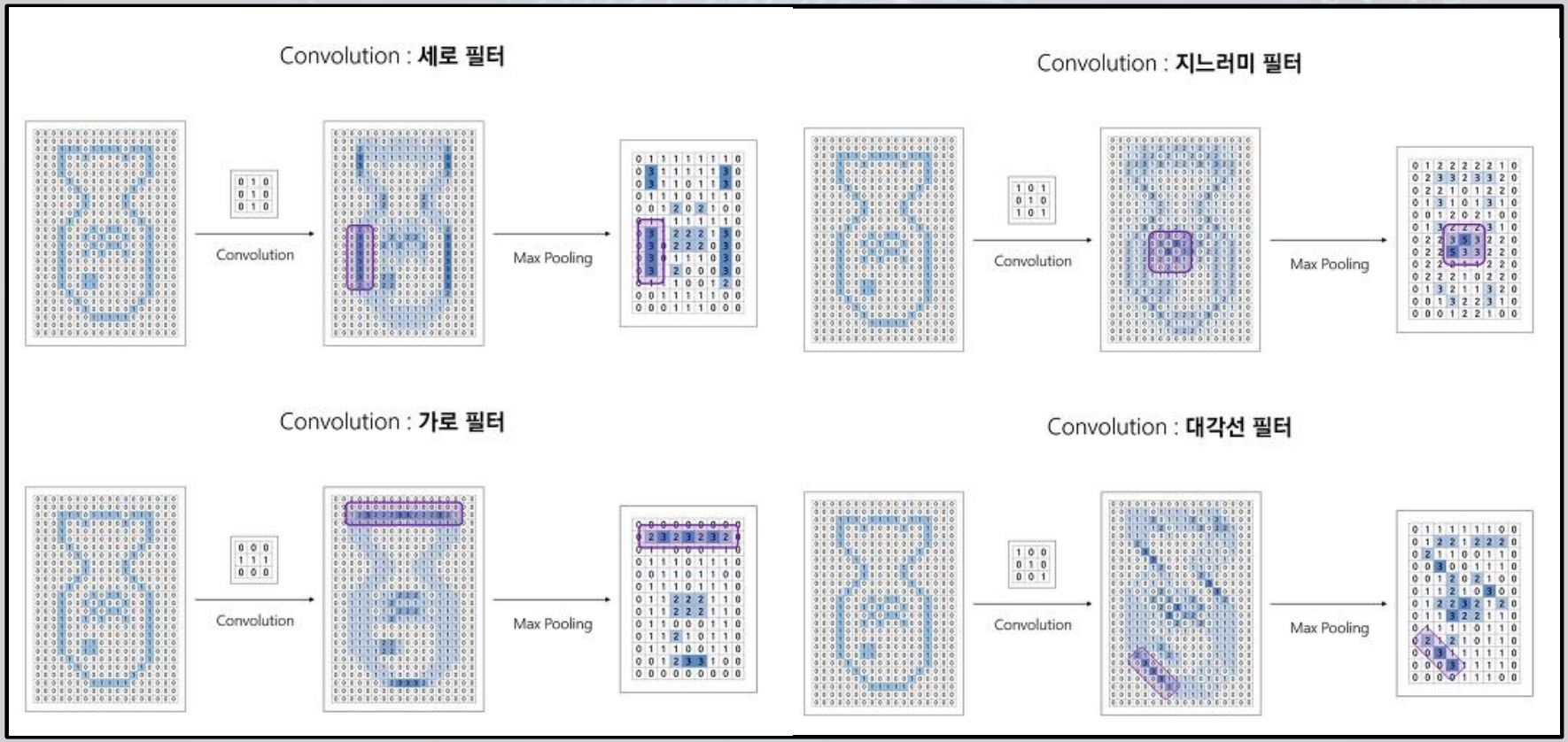
(1x1)+(0x1)+(0x0)+(1x1)=2



(1x1)+(0x1)+(0x1)+(1x0)=1

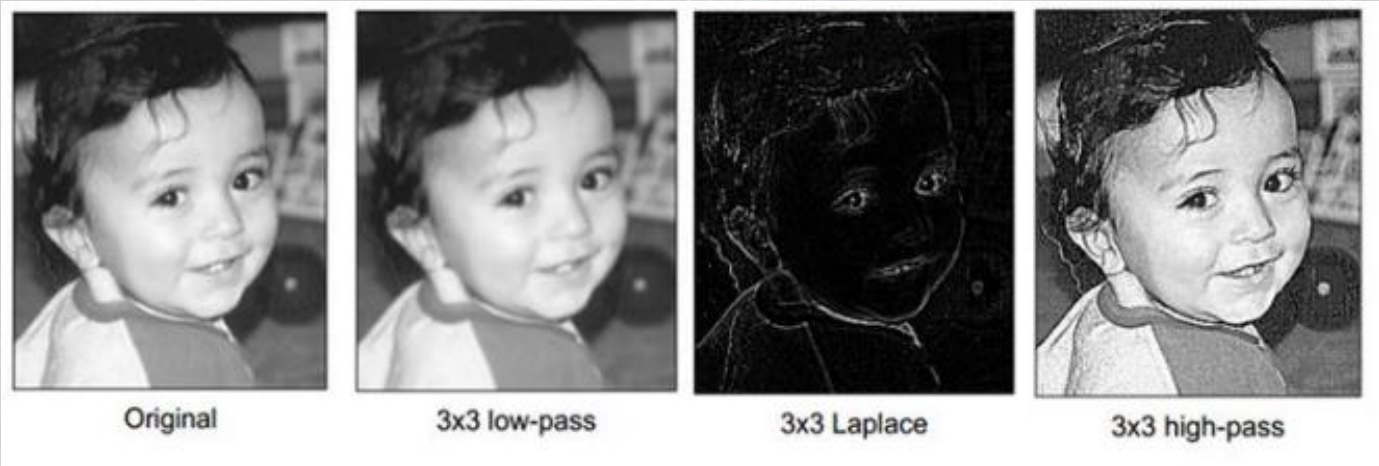
CNN의 구조

- **Convolution** : 다양한 필터 (위아래선, 좌우선, 대각선, 질감1, 질감2, 동그라미, 세모, 네모 등)를 조각 필터로 해당 패턴이 이미지 위에 있는지 확인하여 마킹하는 작업



CNN (Convolution Neural Network)

- 원영상에 대해, 다양한 3*3 filter 연산을 한 경우이며, 필터의 종류에 따라 각기 다른 특징을 끄집어 낼 수 있음

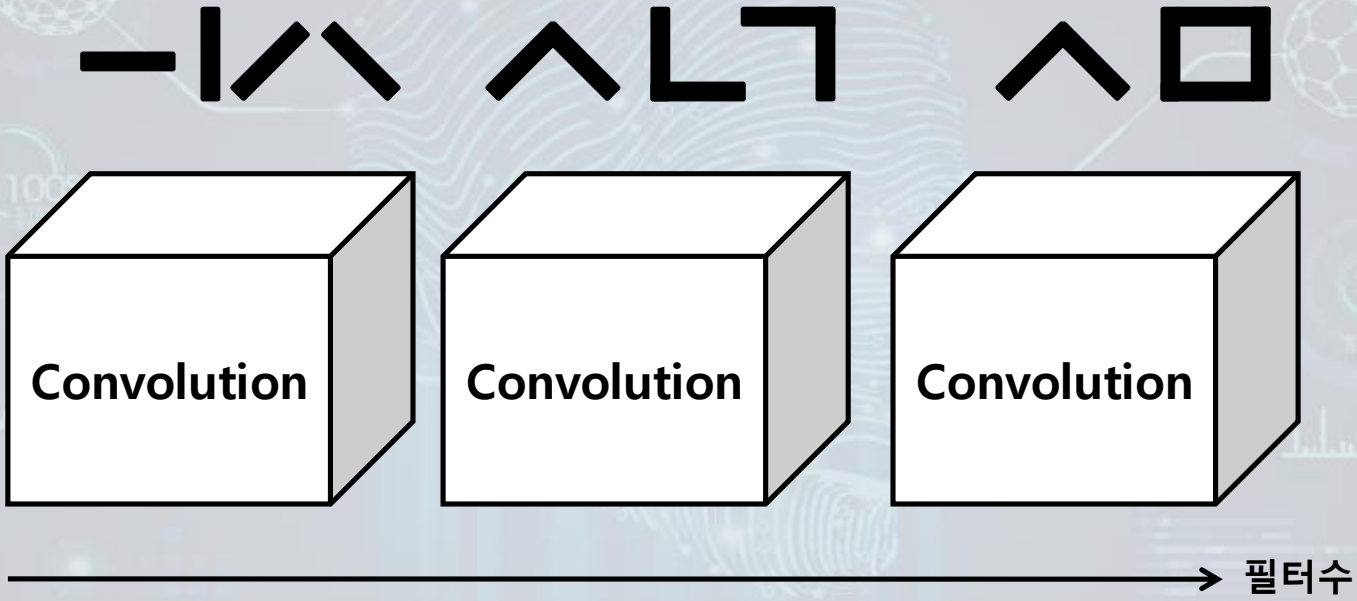


CNN의 구조

Convolution의 멋진점은
간단한 필터를 쌓아가면서 엄청나게 복잡한 필터를 만들어가는 것

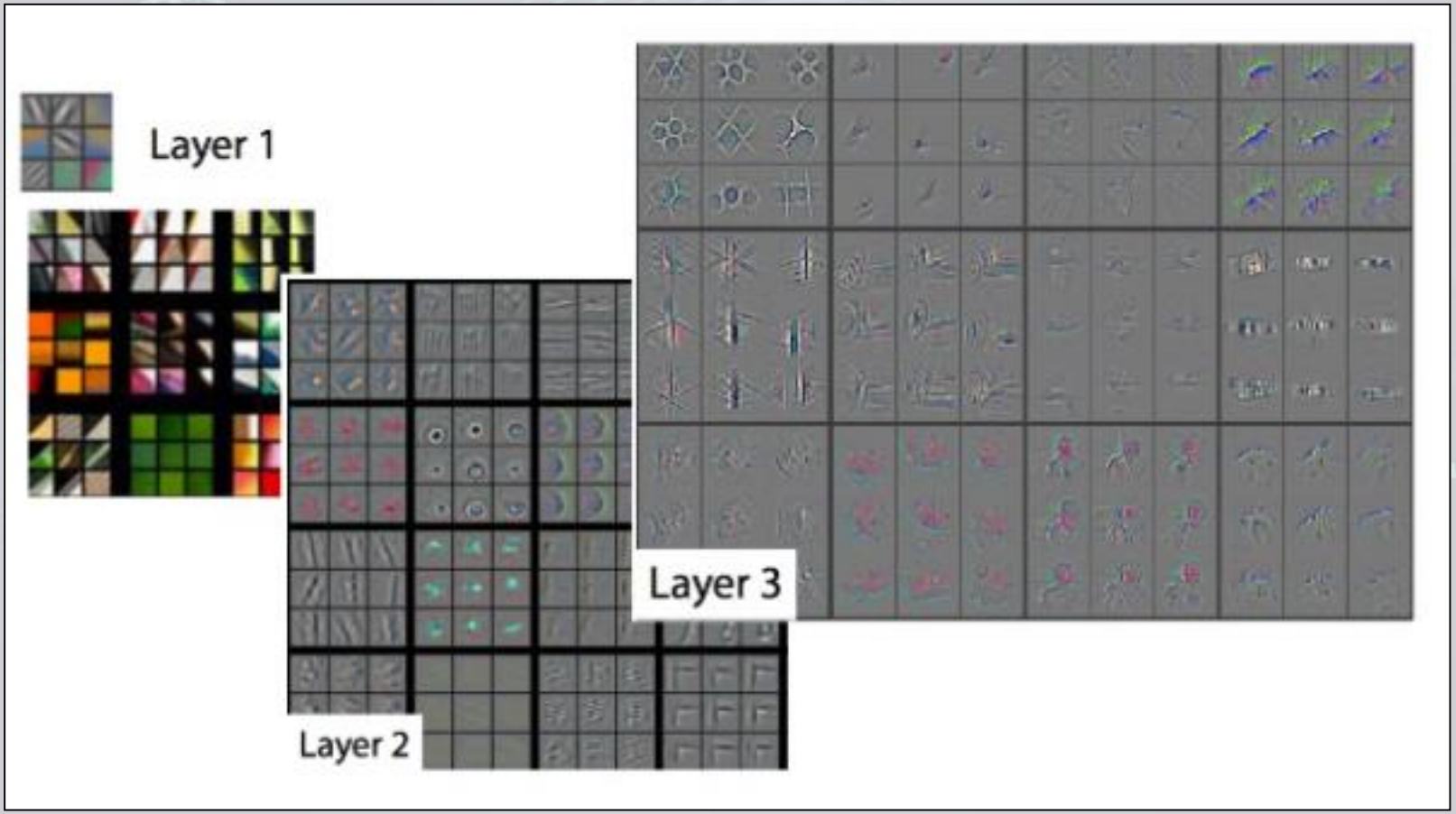
이런 필터를 신경망이 알아서 찾아줘요

CNN의 구조



Convolution은 부품을 조립해서 더 복잡한 부품을 만든다

CNN의 구조



CNN의 구조

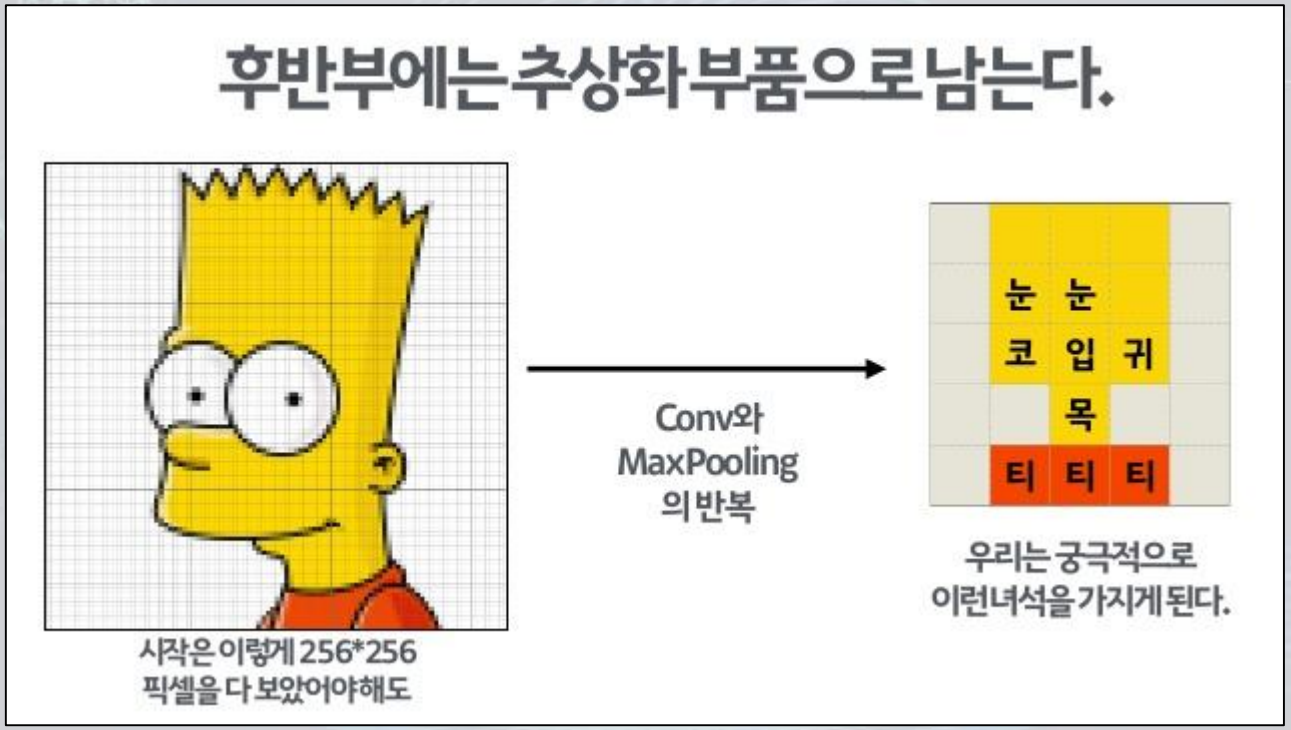
점점 더 멀리서 보는 법 ?

그림의 크기를 줄이면 되겠네요.

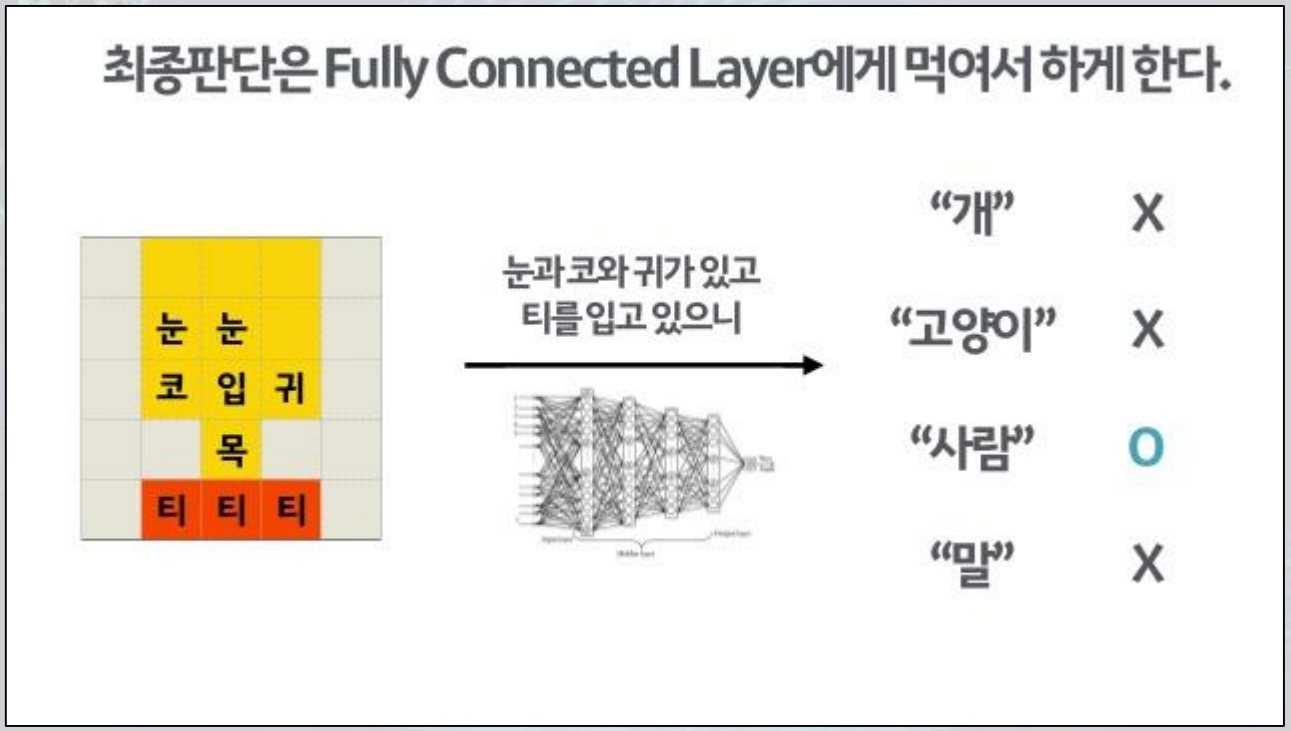
↓

계산량 감소, 노이즈 감소
특징만 추출하여 추론과 학습이 쉬어짐

CNN의 구조


후반부에는 추상화 부품으로 남는다.
시작은 이렇게 256*256
픽셀을 다 보았어야해도
Conv와
MaxPooling
의 반복
눈 눈
코 입 귀
목
티 티 티
우리는 궁극적으로
이런녀석을 가지게 된다.

CNN의 구조


최종판단은 Fully Connected Layer에게 먹여서 하게 한다.
눈 눈
코 입 귀
목
티 티 티
눈과 코와 귀가 있고
티를 입고 있으니
"개" X
"고양이" X
"사람" O
"말" X

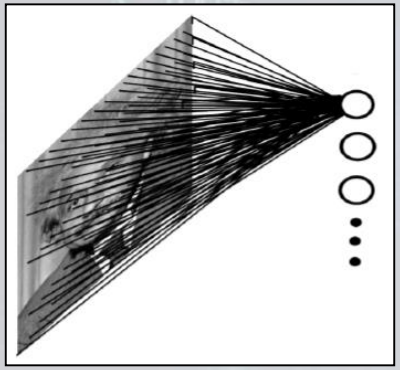
CNN (Convolution Neural Network)

- 특징
  - **Locality(Local Connectivity)**
    - CNN은 Local 정보를 활용하여 인접한 값들에 대한 correlation 관계를 비선형 필터를 적용하여 추출
    - 필터를 여러 개 적용하면 다양한 local 특징 추출 가능
    - 이러한 과정을 거치면서 영상의 크기는 줄어들며 global feature를 추출
  - **Shared weights**
    - Filter를 반복적으로 적용함으로써 변수를 획기적으로 줄일 수 있음
    - Topology(형상) 변화에 무관한 항상성(invariance)를 얻을 수 있음

CNN (Convolution Neural Network)

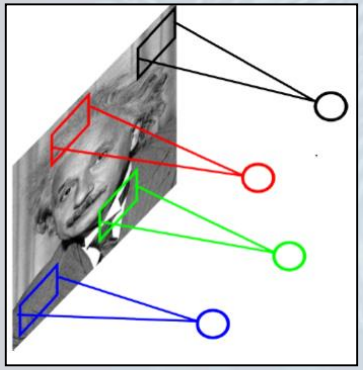
- 전통적인 Neural Networks에서는 모든 output unit이 모든 input unit과 interact 됨
- CNN에서는 특정 영역의 feature와 output이 interact하며 이것을 **sparse interaction**이라고 함
- Sparse interaction은 원본 이미지보다 크기가 작은 filter를 사용함으로써 실현됨

Fully Connected Layer



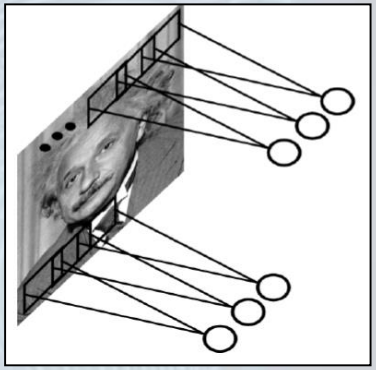
1,000x1,000 이미지
1M hidden unit
10^{12}개의 파라미터

Locally Connected Layer

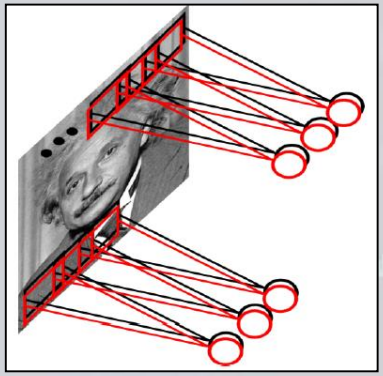


1,000x1,000 이미지
1M hidden unit
필터크기 **10x10**
100M개의 파라미터

Convolutional Layer



한 개의 필터 사용
다른 위치마다 동일한
파라미터를 적용



여러 개의 필터 사용
1,000x1,000 이미지
100개 필터
필터크기 **10x10**
10K개의 파라미터

CNN (Convolution Neural Network)

- 컨볼루션 층 / 맥스 풀링은 2차원 데이터를 사용하므로 데이터를 변환 (28 x 28)

```
19  X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32") / 255
20  X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype("float32") / 255
21
22  Y_train = np_utils.to_categorical(Y_train, 10)
23  Y_test = np_utils.to_categorical(Y_test, 10)
```

CNN (Convolution Neural Network)

- **Conv2D()** : Keras에서 컨볼루션 층을 추가하는 함수

```
4   from keras.layers import Dense, Conv2D
...
19  model = Sequential()
20  model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1),
    activation="relu"))
```

- 첫 번째 파라미터 : 마스크(커널, 필터)의 개수
- kernel_size : 마스크의 크기 (행 크기, 열 크기)
- input_shape : 맨 처음 입력되는 값 (행 크기, 열 크기, 색상 (칼라 3, 흑백 1))

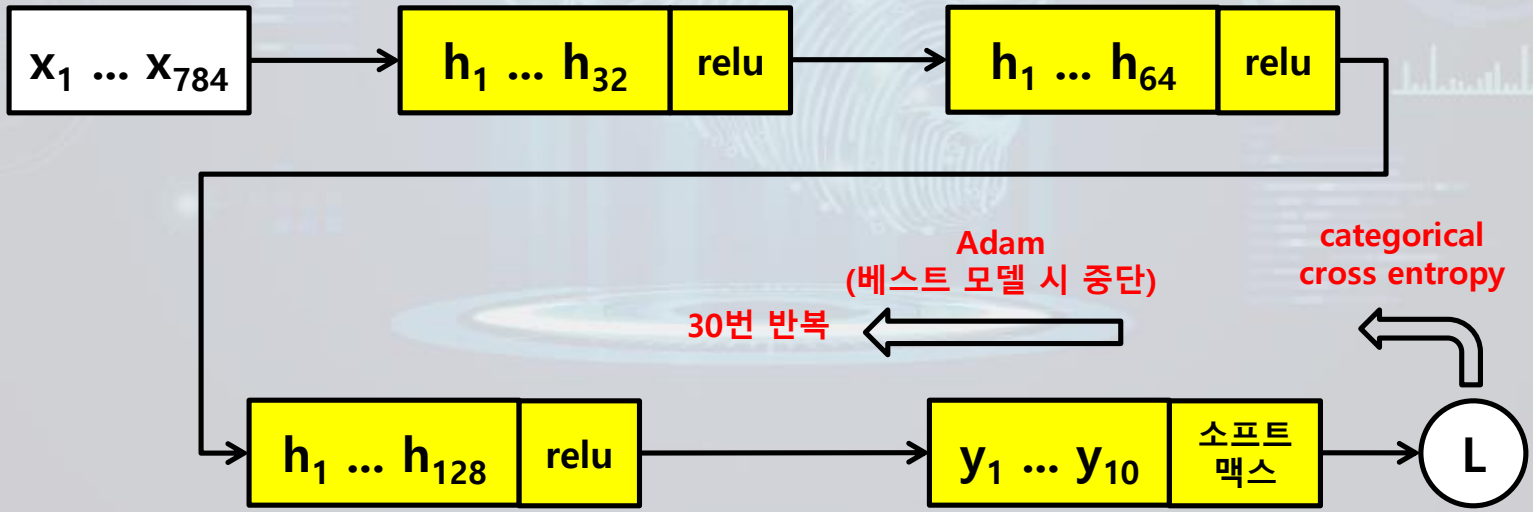
- 컨볼루션 은닉층 추가

```
19  model = Sequential()
20  model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1),
    activation="relu"))
21  model.add(Conv2D(64, (3, 3), activation="relu"))
```

CNN (Convolution Neural Network)

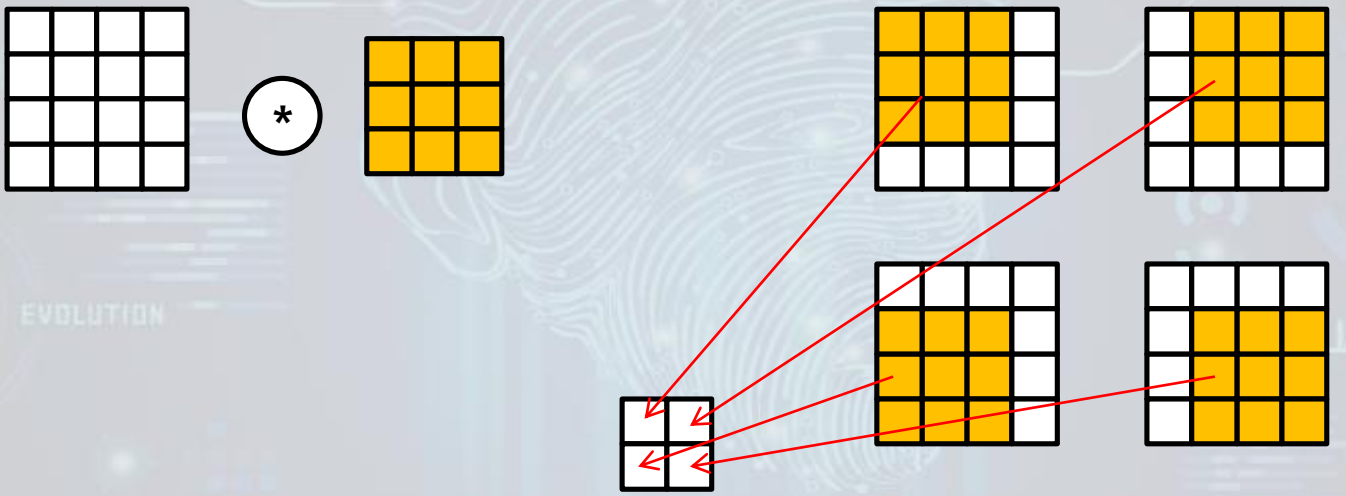
- 컨볼루션 층의 구조

```
19  model = Sequential()
20  model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1),
    activation="relu"))
21  model.add(Conv2D(64, (3, 3), activation="relu"))
22  model.add(Flatten())
23  model.add(Dense(128, activation="relu"))
24  model.add(Dense(10, activation="softmax"))
```



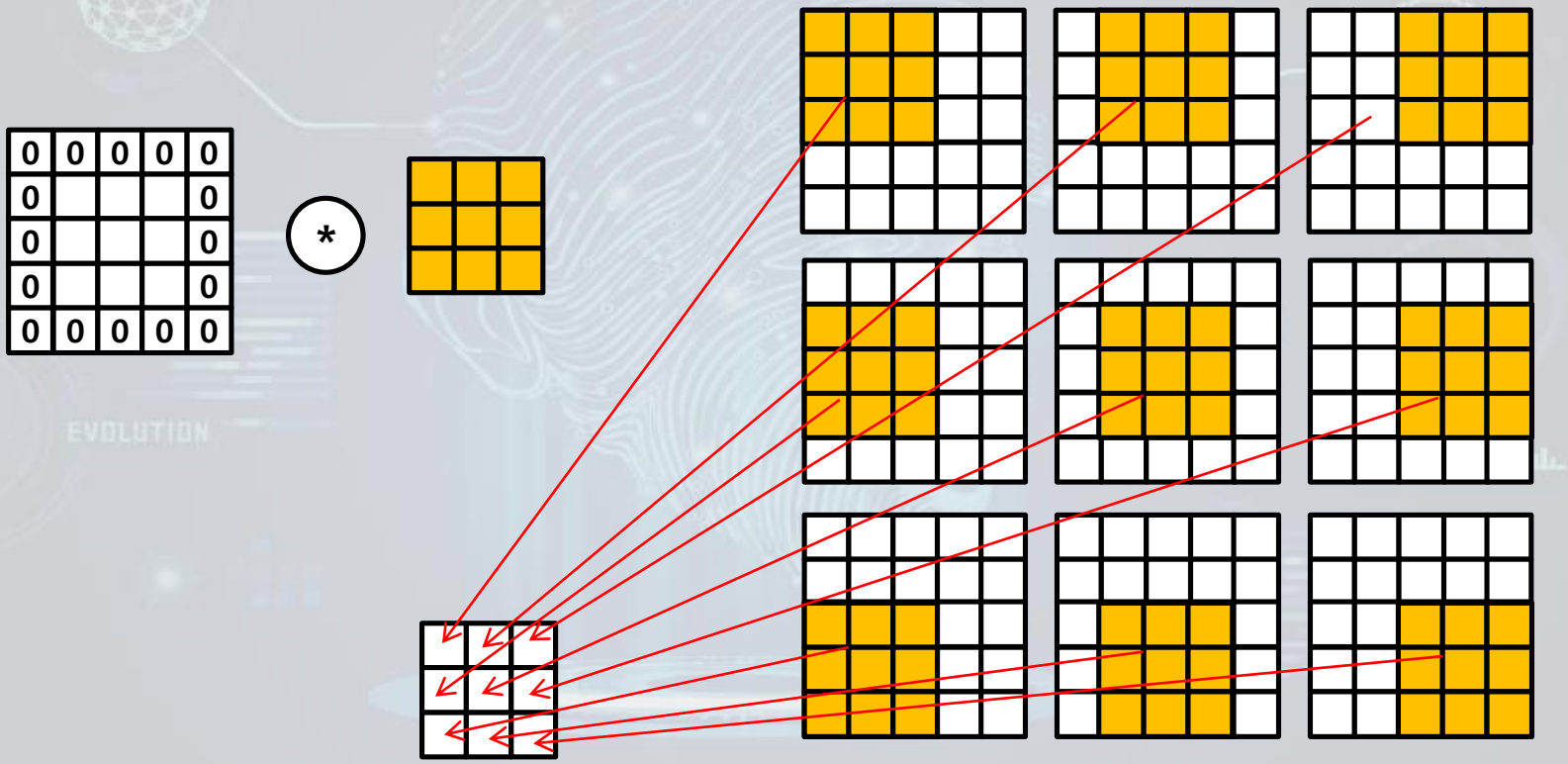
CNN의 구조

- **Zero padding** : 패딩을 사용하지 않으면 층을 지날 때마다 크기가 줄어들므로 이러한 문제를 해결하기 위해 테두리 부분을 컨볼루션 전에 0으로 채우는 작업

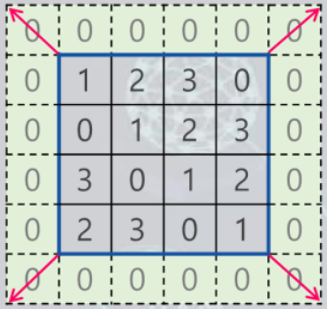


CNN의 구조

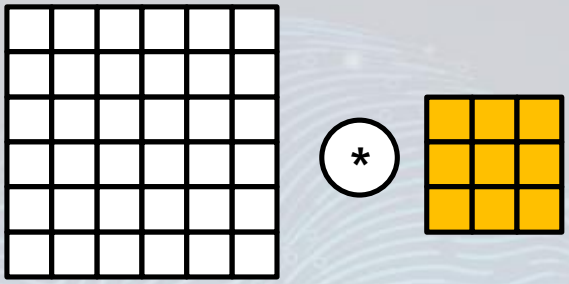
- Zero padding을 적용한 결과

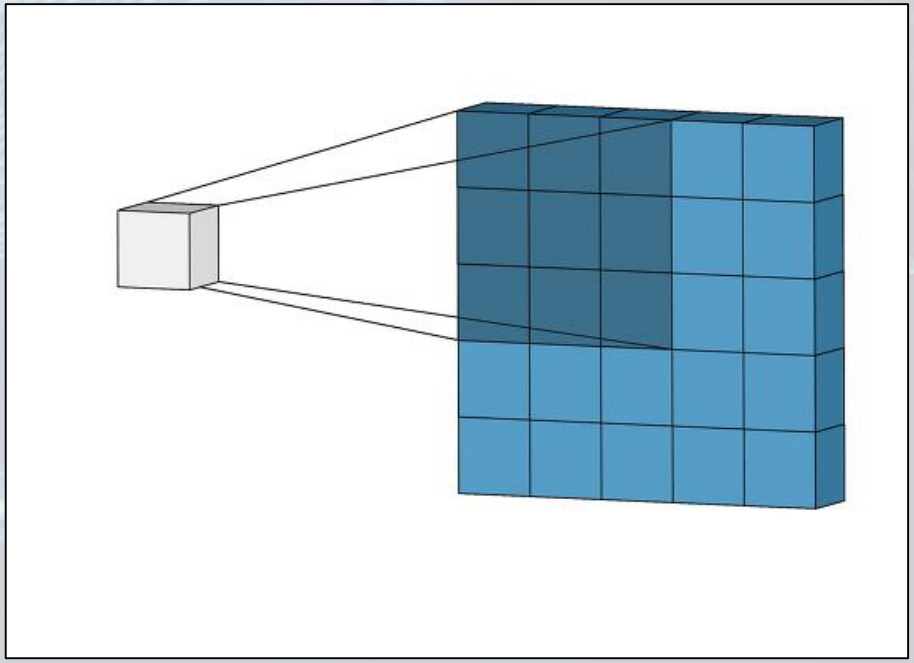


CNN의 구조



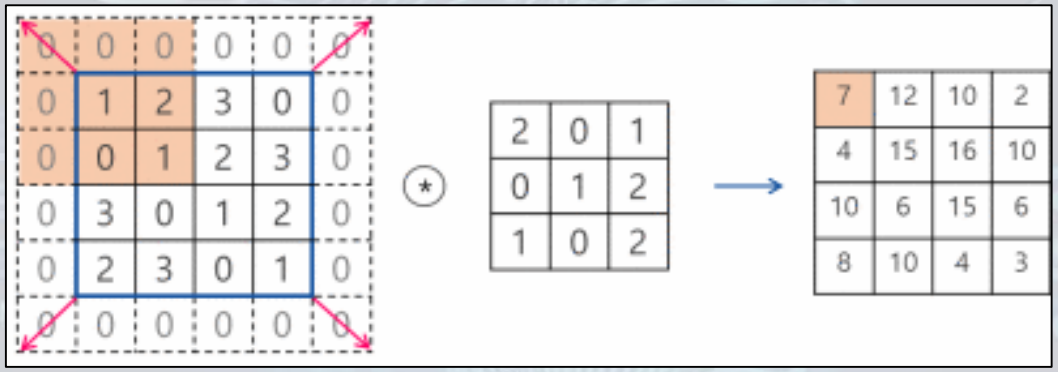
1폭짜리 zero- padding





CNN의 구조

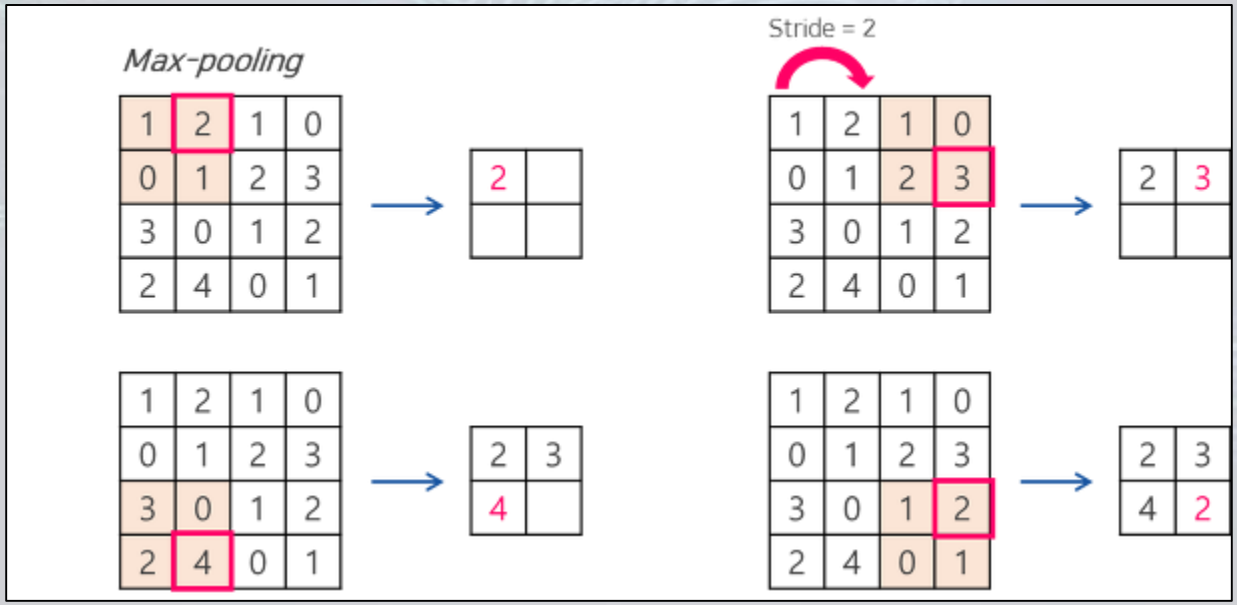
- **Stride** : 입력 데이터에 필터를 적용할 때 이동할 간격을 조절하는 것 (필터가 이동할 간격)



zero-padding과 Stride를 1로 적용한 경우 컨볼루션

CNN의 구조

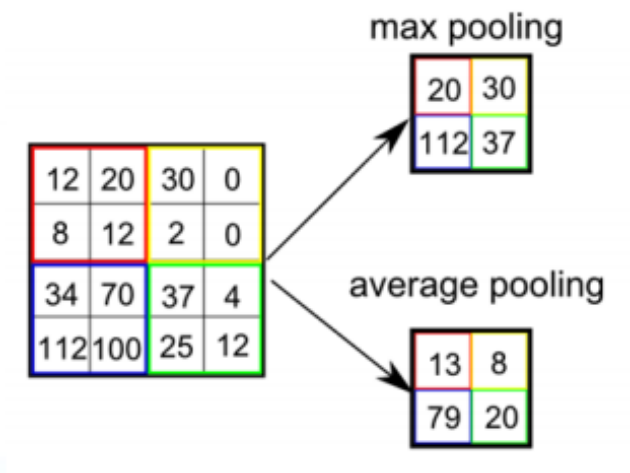
- Pooling (Sub-Sampling) : 처리할 데이터의 크기를 줄이는 작업 (Max-Pooling, Average-Pooling)
  - Max-Pooling : 최대값을 대표값으로 설정
  - Average-Pooling : 평균값을 대표값으로 설정
  - L2-norm pooling : L2 규제에 의한 값을 대표값으로 설정



Stride를 2로 설정한 경우 Max-Pooling

CNN (Convolution Neural Network)

- **Pooling (Sub-Sampling)**
  - 단순히 데이터의 사이즈를 줄여주며, 노이즈를 상쇄시키고, 미세한 부분에서 일관적인 특징을 제공
  - 보통은 Convolution과정에서 만들어진 feature들의 가장 큰 값들만 가져와 사이즈를 줄임 → **Max-pooling**



CNN (Convolution Neural Network)

- 단점

(1) 메모리 관점에서 일반적인 다층 퍼셉트론보다 더 많은 용량을 차지 (많은 파라미터)

(2) 실행 시간 관점에서 컨볼루션 과정이 많은 계산을 필요로 하고 전체 실행 시간 중 약 60%이상을 차지

(3) 같은 개수의 파라미터를 갖는 신경망보다 약 3배 가까이 실행 시간이 느림

CNN (Convolution Neural Network)

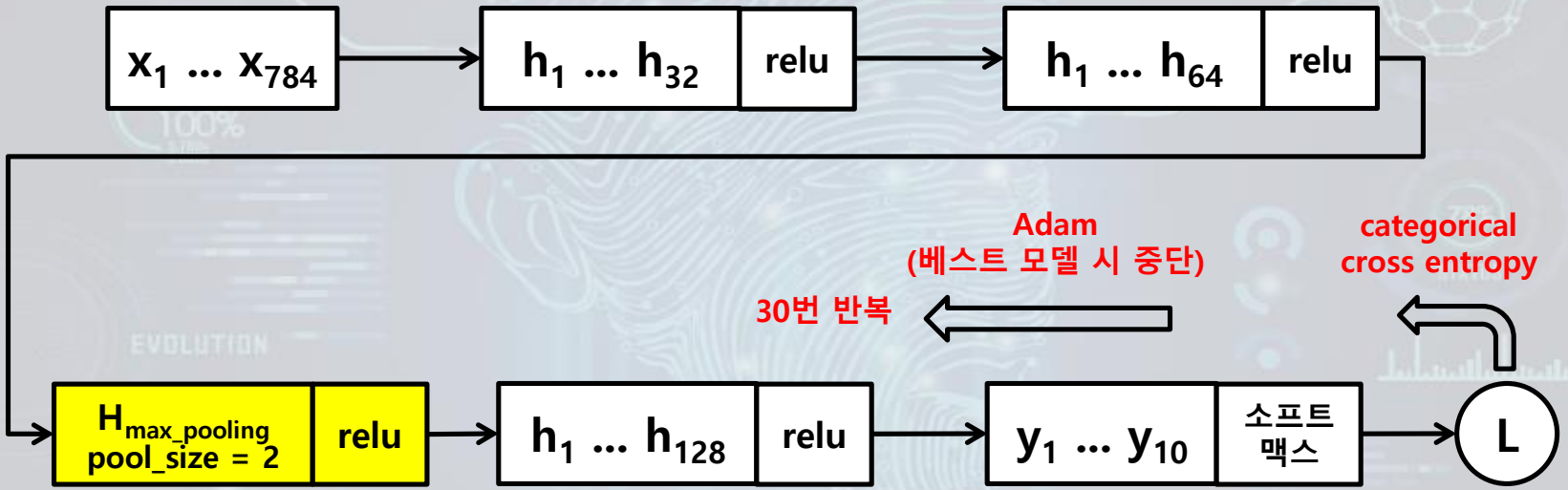
- Max Pooling

```
4   from keras.layers import Dense, Conv2D, MaxPooling2D
... ...
19  model = Sequential()
20  model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1),
    activation="relu"))
21  model.add(Conv2D(64, (3, 3), activation="relu"))
22  model.add(MaxPooling2D(pool_size=2))
23  model.add(Dense(128, activation="relu"))
24  model.add(Dense(10, activation="softmax"))
```

- MaxPooling2D() : Max Pooling을 구현해주는 함수
- pool_size : 풀링 창의 크기를 설정하는 것으로 2로 설정하면 전체 크기가 ½로 줄어듬
  (2, 2)로 하면 ¼로 줄어듬

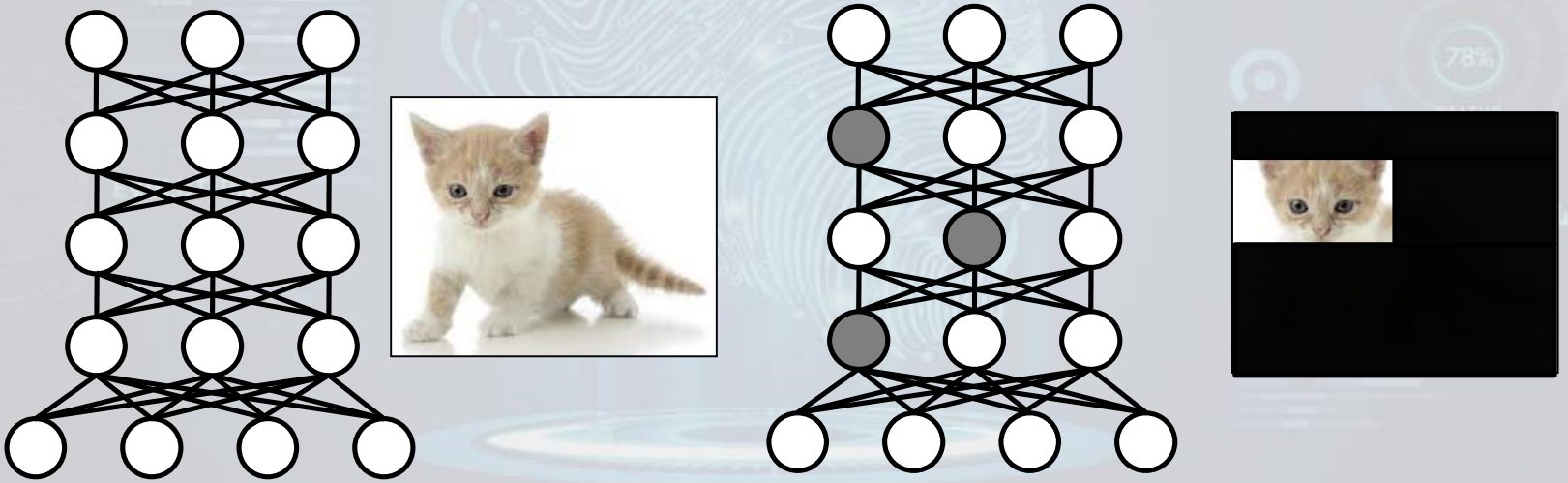
CNN (Convolution Neural Network)

- Max Pooling



CNN (Convolution Neural Network)

- Drop out
  - 노드가 많아지거나 층이 많아진다고 해서 학습이 무조건 좋아진다고 볼 수 없음 → 과적합
  - 과적합을 해결하는 가장 좋은 방법 → Drop out (은닉층에 배치된 노드 중 일부를 사용하지 않게 하는 방법)



CNN (Convolution Neural Network)

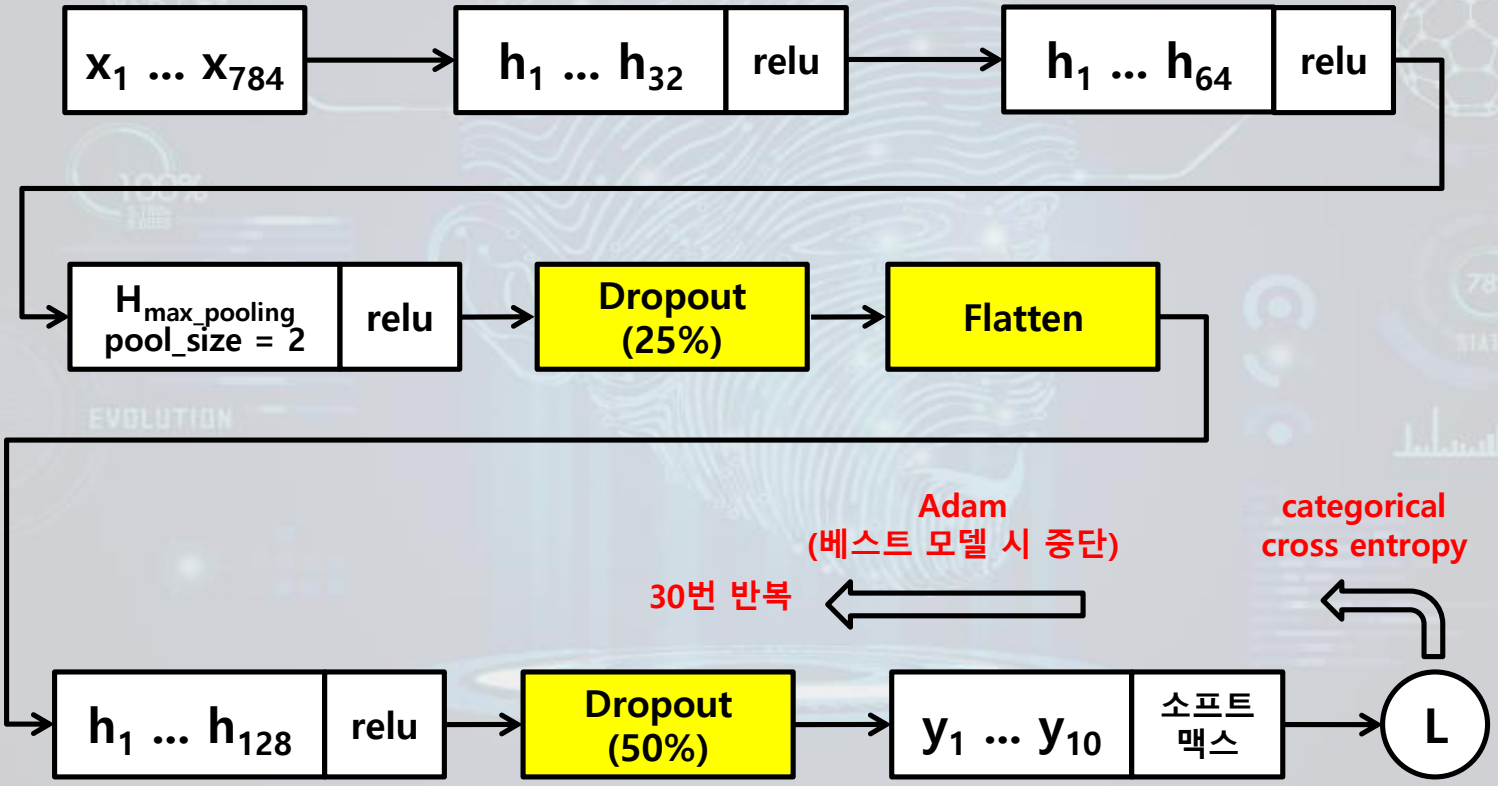
- Drop out

```
4   from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
... ...
19  model = Sequential()
20  model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1),
    activation="relu"))
21  model.add(Conv2D(64, (3, 3), activation="relu"))
22  model.add(MaxPooling2D(pool_size=2))
23  model.add(Dropout(0.25))
24  model.add(Flatten())
25  model.add(Dense(128, activation="relu"))
26  model.add(Dropout(0.5))
27  model.add(Dense(10, activation="softmax"))
```

- Dropout(0.25) : 25%의 노드를 drop 시킴
- Flatten() : 컨볼루션 층이나 팩스 풀링은 2차원 데이터를 다루므로 Dense()와 연결하기 위해 1차원으로 변환하는 기능

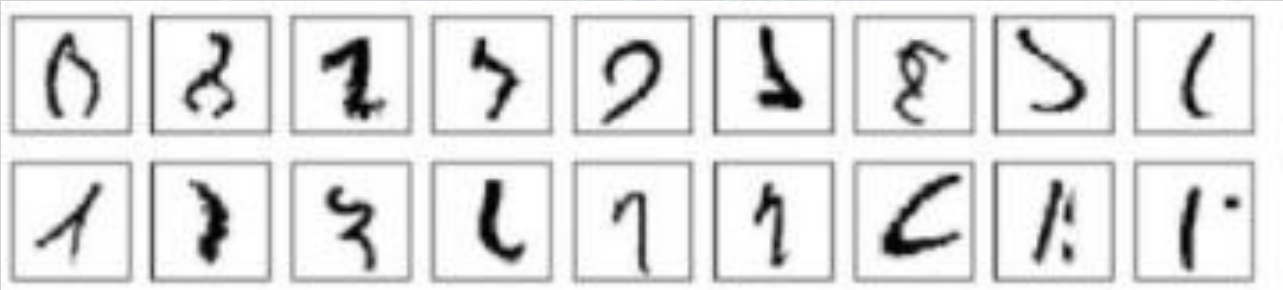
CNN (Convolution Neural Network)

- Drop out



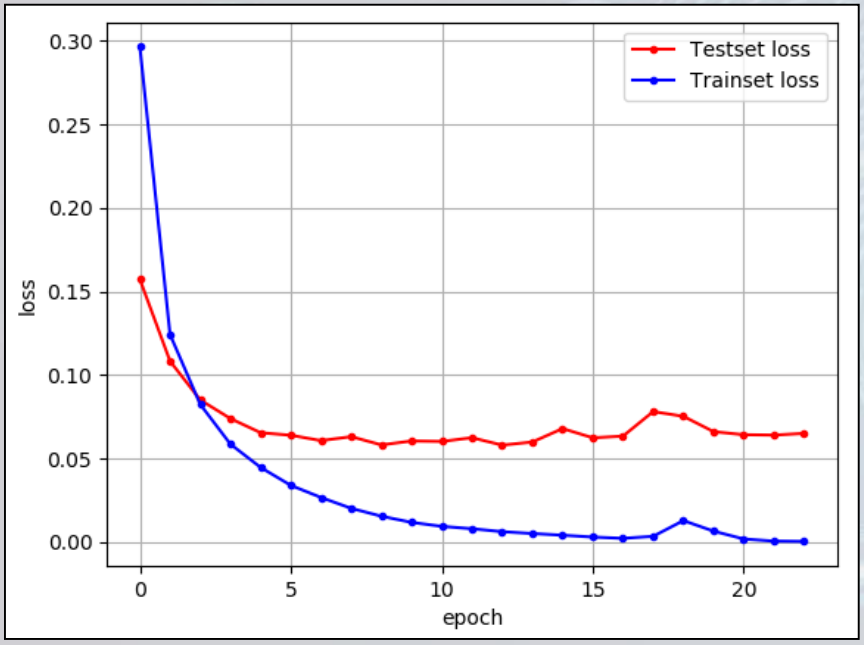
CNN (Convolution Neural Network)

- 알아내지 못한 숫자의 예

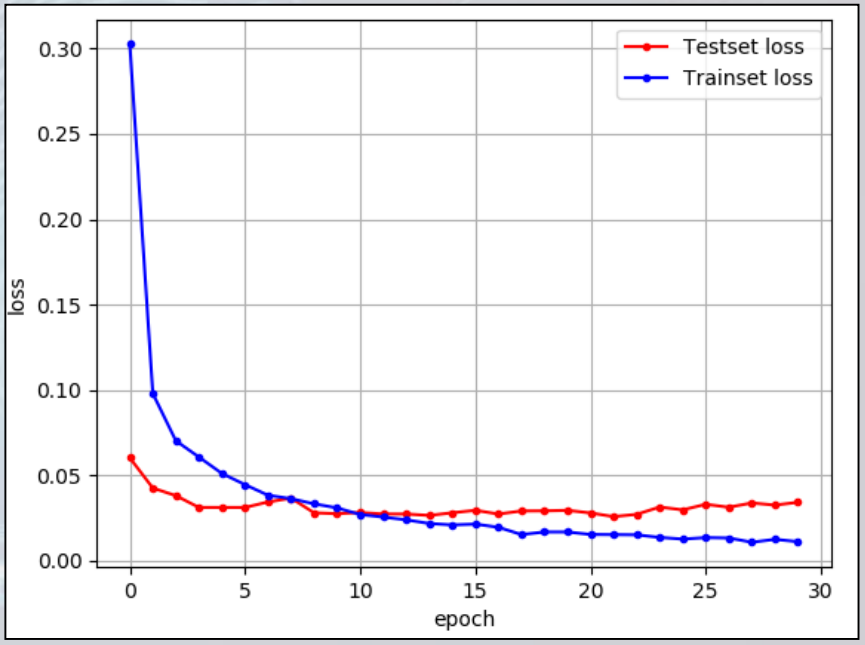


CNN (Convolution Neural Network)

- 실행 결과

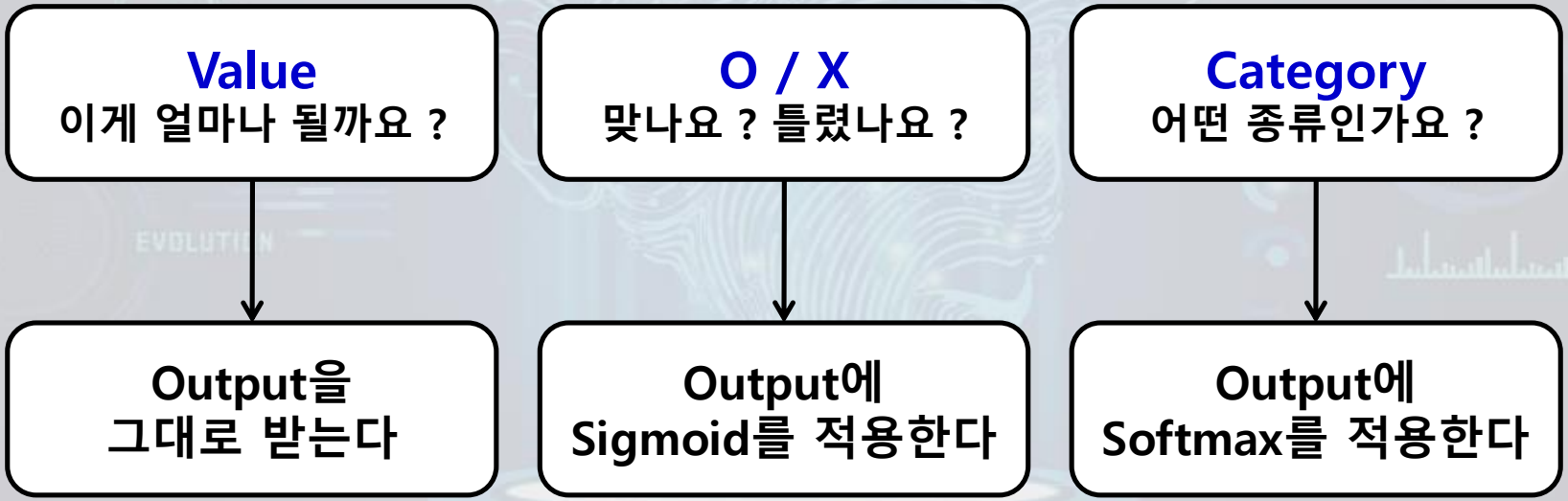


이전의 결과



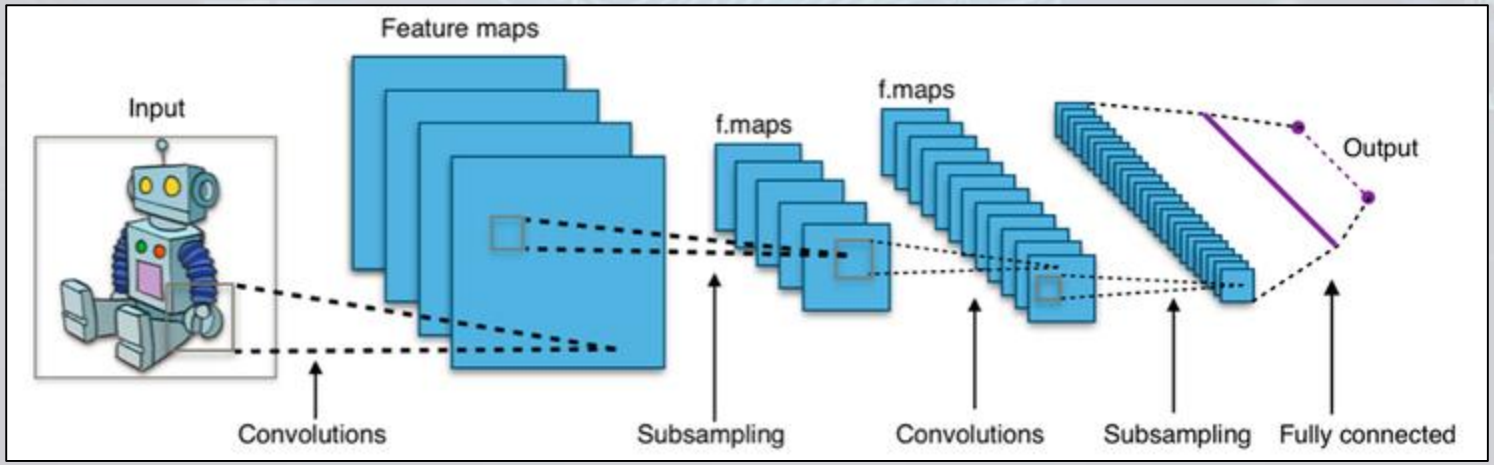
CNN의 구조

신경망의 답을 받는 3가지 방법



CNN (Convolution Neural Network)

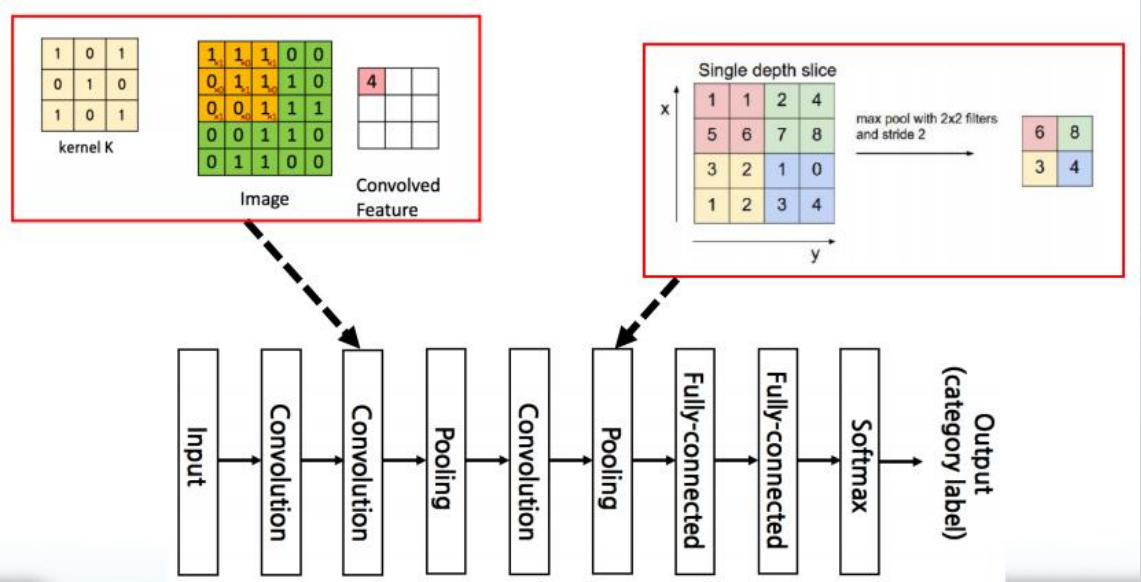
- 기존의 방식은 **데이터에서 지식을 추출해 학습**이 이루어졌으나, CNN은 데이터를 **feature(특징, 차원)로 추출**하여 이 **feature들의 패턴을 파악**하는 구조
- CNN알고리즘은 Convolution 과정과 Pooling 과정을 통해 진행
- CNN은 Convolution layer, Pooling layer, fully-connected layer로 구성됨
- CNN의 활용 : 정보추출(Information Extraction), 문장분류(Sentence Classification), 얼굴인식(Face Recognition) 등



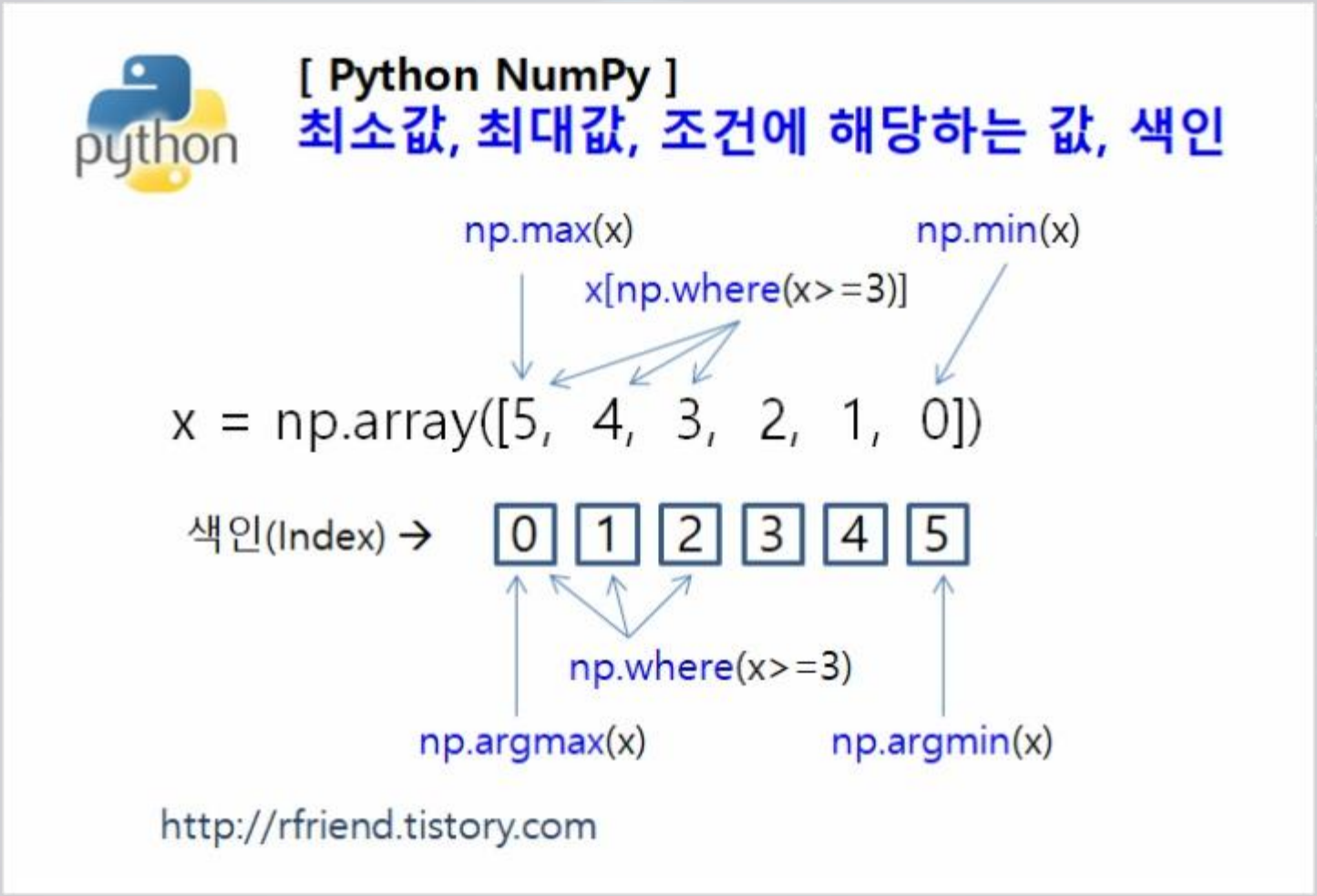
http://physics2.mju.ac.kr/juhapruwp/?p=1517

CNN (Convolution Neural Network)

- CNN Architecture
  - Input layer 이후에 convolution layer와 pooling layer가 반복되고 마지막 1 또는 2 layer에서 fully-connected layer로 구성
  - Convolution layer는 filter의 명암 패턴과 유사한 패턴이 입력된 이미지가 어디에 있는지 검출
  - Pooling layer는 overfitting을 막기위해 convolution layer의 출력 이미지의 특징은 살리고, 크기는 줄임

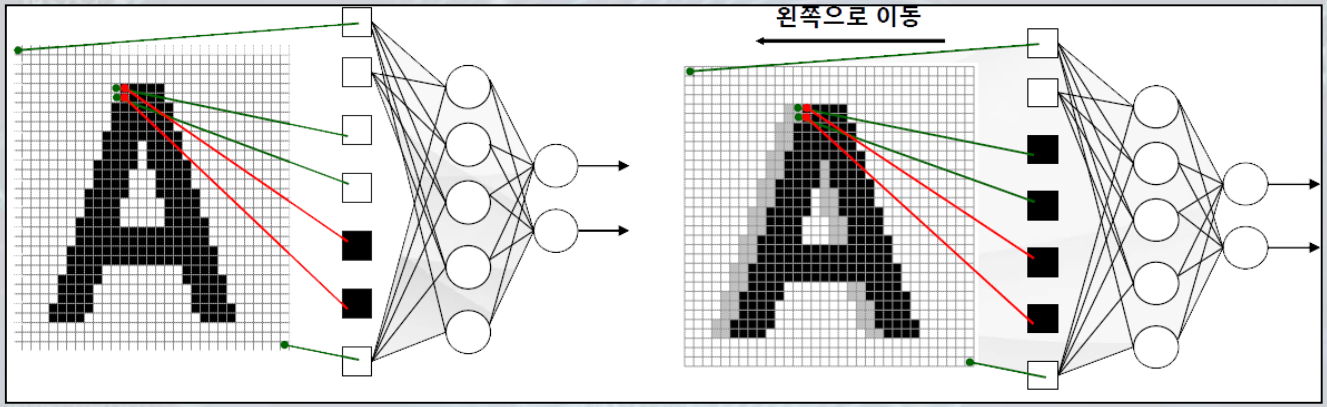


argmax()와 argmin()



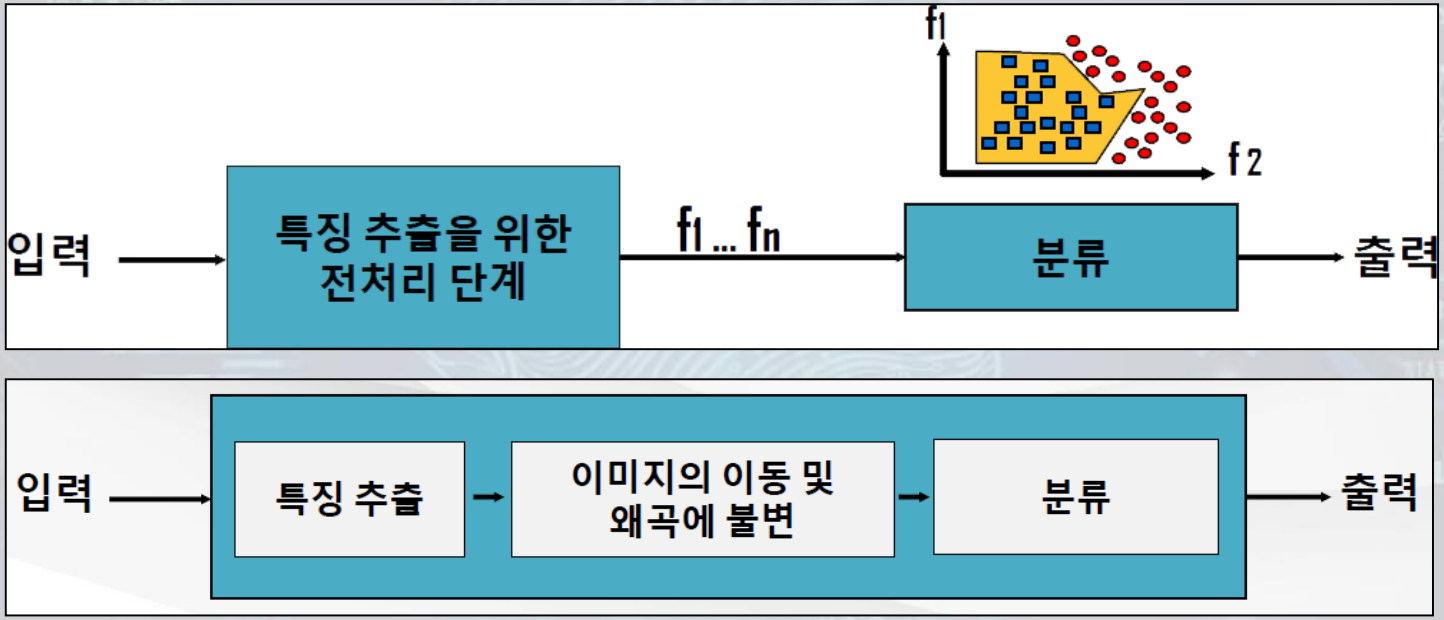
CNN (Convolution Neural Network)

- 일반 신경망은 이지미의 위치, 크기, 각도 변화 등에 취약함



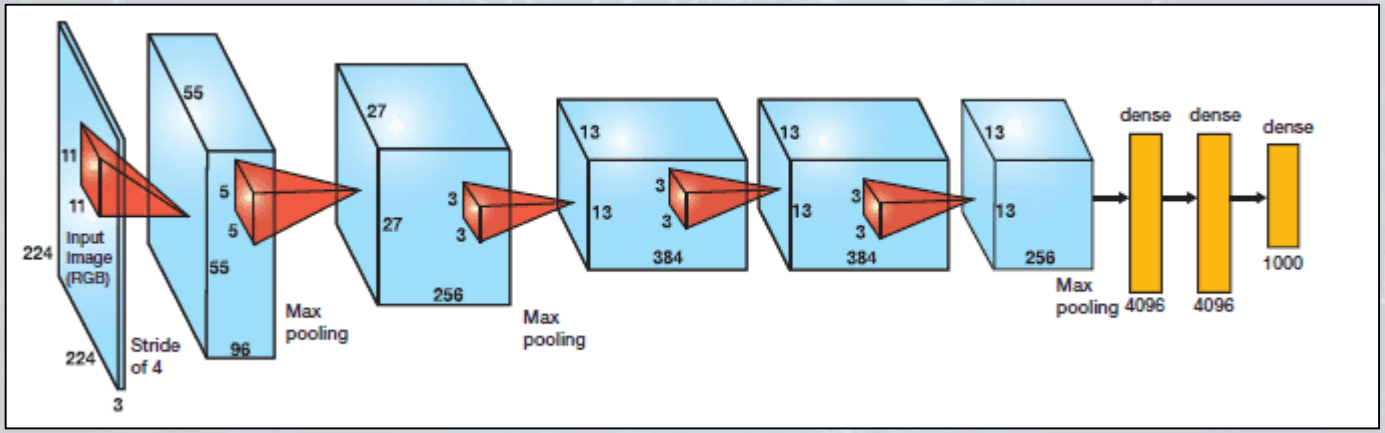
CNN (Convolution Neural Network)

- 일반 분류기와 CNN



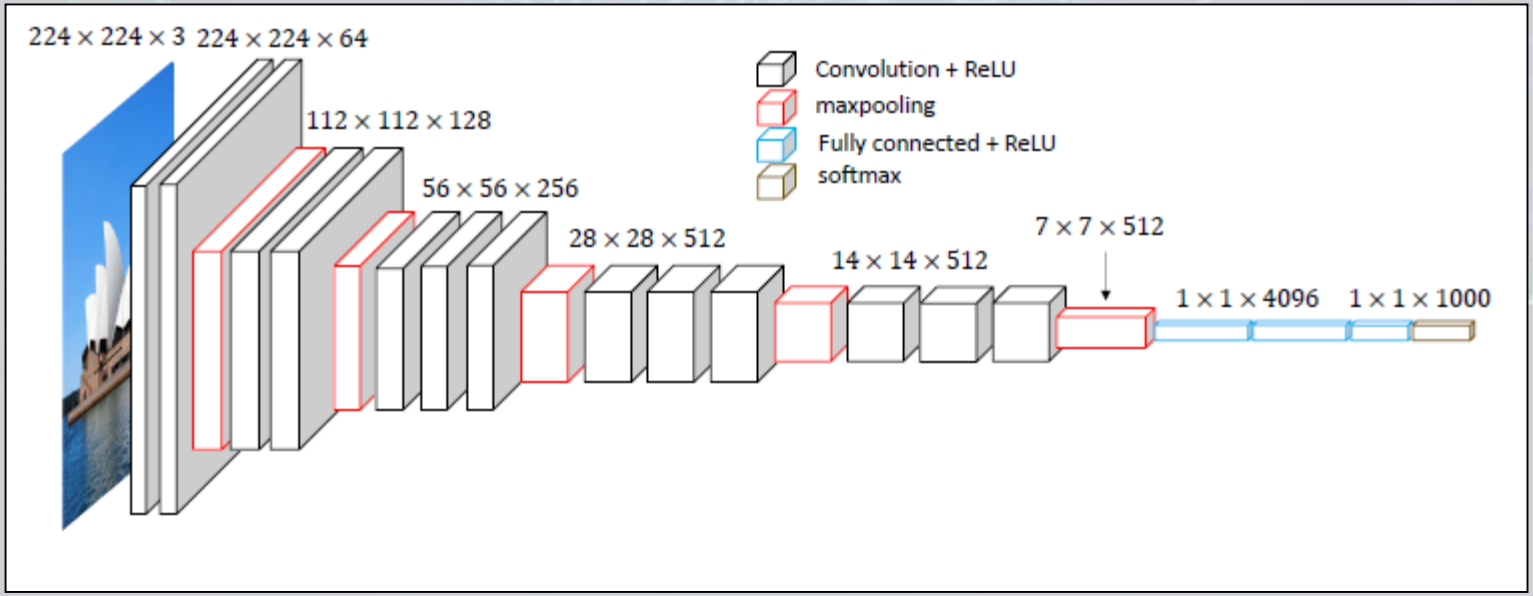
CNN (Convolution Neural Network)

- **AlexNet**의 구조 : 5개의 convolutional Layers, 3개의 Maxpooling Layers, 3개의 Fully Connected Layers



CNN (Convolution Neural Network)

- **VGG16** 구조 – 적당히 복잡한 데이터에 적합한 네트워크 구조 (컨볼루션 층 13개, Maxpooling 층 5개, Fully Connected Layers 3개로 구성)
- VGG-CNN-M, LeNet과 본질적으로 거의 비슷한 구조



CNN (Convolution Neural Network)

- **GooleNet**의 구조 : 동일한 구조를 갖는 building block (**Inception**)을 통해 매우 깊은 네트워크를 설계
- 일반적인 CNN은 네트워크가 깊어지면 학습 파라미터 증가로 연산량 증가 → 깊은 네트워크에서 **NIN (Network In Network)**을 사용하여 Vanishing Gradient, Overfitting, 연산량 증가 문제 해결