CNN

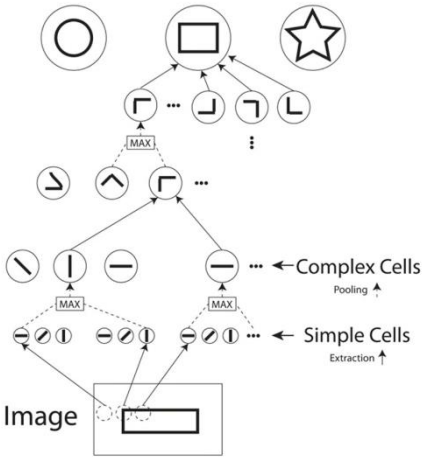
하정욱 교수

Office: CY관 502호

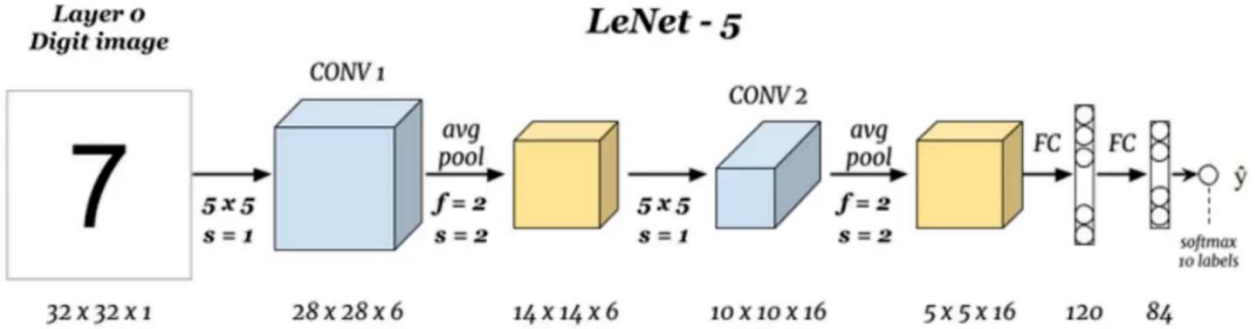
Email: jwha6169@sogang.ac.kr

CNN(Convolutional Neural Network)

- **1989년 안 르쿤 (Yann Lecun) 교수 발표**
  - LeNet-5:
    " Gradient- Based Learning Applied to document Recognition"
- CNN도 사람의 시각 피질을 참고하여 만들었음
  : 뇌의 시각 피질은 서로 계층적으로 연결되어 있음
- **시각적 이미지 분석에 일반적으로 사용**
  - 이미지로 부터 특징을 추출하여 이미지 클래쓰 분류

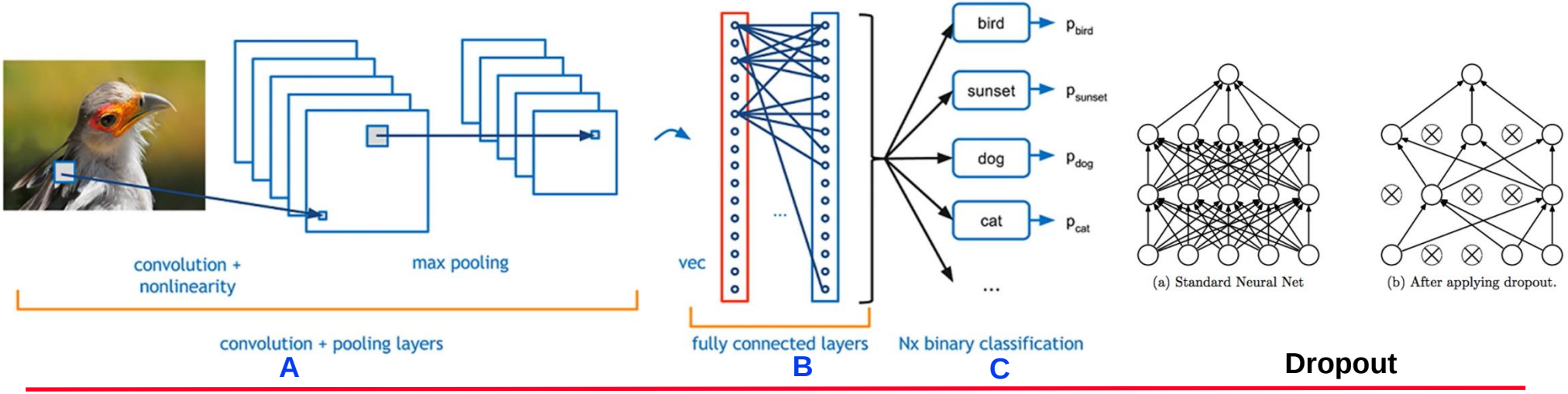


- **이미지 및 비디오 인식, 추천 시스템, 이미지 분류, 의료 이미지 분석 및 자연어 처리에 응용**
  - 이미지 초해상, 이미지 합성, 이미지 컬러화, 사물 탐지 등으로 확대 중
- **일반 신경망의 경우 이미지 자체를 하나의 데이터로 입력하기 때문에 이미지의 특징을 찾지 못하고 이미지의 위치가 약간 변경되거나 왜곡된 경우 성능 저하**
  : 공간적 정보(Topology)가 손실, Spatial Structure 정보 유실, Overfitting 발생



CNN(Convolutional Neural Network)

- CNN은 이미지에서 객체, 얼굴, 장면을 인식하기위해 패턴을 찾는데 유용한 알고리즘
  - 기존 방식은 데이터에서 지식을 추출해 학습을 이루었지만, CNN은 데이터의 특징을 추출하여 특징의 패턴을 파악하는 구조
  - 낮은 층에서는 단순한 패턴을 높은층으로 올라갈수록 복잡한 패턴 인식
  - 이미지를 하나의 데이터가 아닌 여러 개로 분할하여 처리-> Locality
  - 이미지가 왜곡되더라도 이미지의 부분적 특성을 추출할 수 있어 성능을 낼 수 있음
- 활성화 함수 : ReLU, Softmax, cross-entropy loss
- Dropout
  - 은닉층의 일부 Node를 없애면서 성능 저하는 최소화하고 Overfitting 방지
  - Co-adaptation 보완, 앙상블 처리도 가능

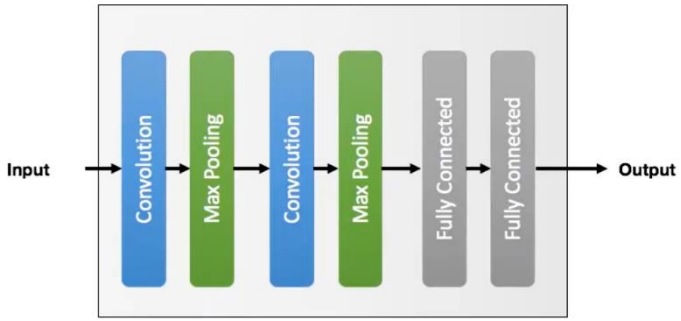


CNN – Convolution Layer

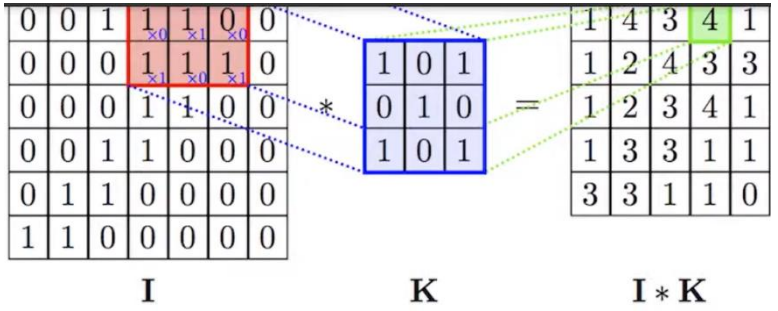
- 합성곱층(Convolution Layer)
  - 입력데이터에 필터를 통한 Convolution 연산을 하여 총합을 구함
    : 필터수 만큼 특성맵(Feature map, Activation map, Response map)) 도출
  - Filter는 입력 데이터와 상관없이 일정한 학습 파라미터 개수를 가지고 있어 과적합 방지
  - 3차원 이미지 데이터에 대해 필터도 이미지와 같은 채널수를 갖고 있어야 함
  - 공간 정보 유지하면서 인접 이미지와 특징을 효과적으로 인식

고해상도 원본이미지	*	Convolution Feature	=	Feature map (이미지 윤곽)

- 합성곱 연산 후 데이터 사이즈
  (n– f+1) * (n– f+1) 단, n : 입력 데이터의 크기, f: 필터(커널의 크기)



Basic Convolution Neural Network



Filter를 통한 합성곱 연산

CNN- Convolution 연산

- 입력데이터가 들어오면 필터의 윈도우가 일정 간격으로 이동해가며 합성곱(내적) 연산
- CNN은 학습이 거듭되며 Filter의 가중치와 편향을 매번 갱신되는 것
- Filter의 개수 만큼 특성맵이 나옴

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

*

Convolution Feature (Filter) = Red

1	0	0
0	1	1
1	0	1

1*1+0*0+0*0=1
1*0+1*1+1*1=2
0*1+1*0+1*1= 1

1+2+2=4

R(Red 추출)

4	3
4	4

Convolution Feature = Blue

1	0	1
0	1	0
0	1	1

B(Blue추출)

1	1
2	1

Convolution Feature = Green

1	0	1
0	0	1
0	1	1

G(Green 추출)

2	1
1	3

CNN알고리즘- Convolution

예) 컬러색이므로 RGB(Red, Green, Blue)로 나누어서 계산 후 처리

- Convolution feature가 다른 이유는 가중치가 다르기 때문-> 학습하며 자동으로 추출
  - Red, Green, Blue에서 얻은 Feature Map 을 더함
  - 새로운 이미지의 마지막 차원수는 필터의 수와 동일

- **합성곱 연산의 결과가 활성화 함수 ReLU를 지남 : Convolution Layer**

Feature Map(Red)

4	3
4	4

+

Feature Map(Green)

2	1
1	3

+

Feature Map(Blue)

1	1
2	1

7	5
7	8

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

7	5
7	8

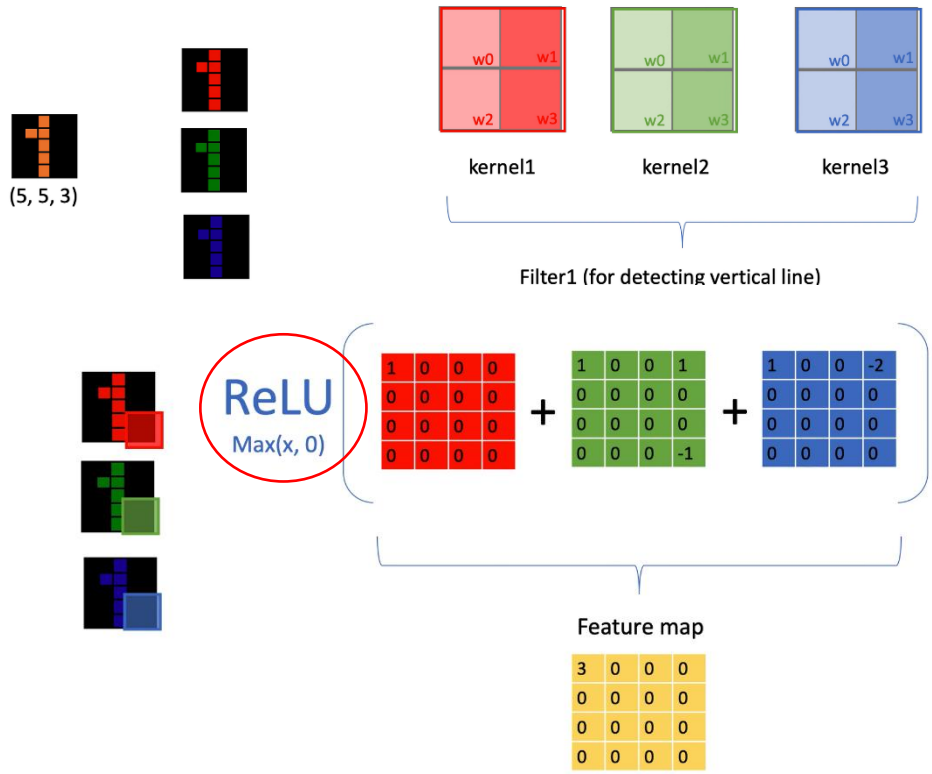
CNN - Filter

- **Filter VS. Kernel**
  - 주어진 데이터에 특정한 특징이 있는지 없는지 검출해주는 함수
  - **CNN**에서 학습의 대상
  - 학습된 필터들은 경계선을 찾거나 블러리한 면을 찾는 등 다양한 주파수필터의 기능 수행
  - 출력에 가까운 레이어일수록 추상화 수준이 높아 해석이 어렵다
  - 복수의 필터로 이미지 특징 추출 학습
  - 입력 데이터의 채널수와 필터의 수는 같다

  : 각각의 필터는 각각 다른 가중치를 가짐

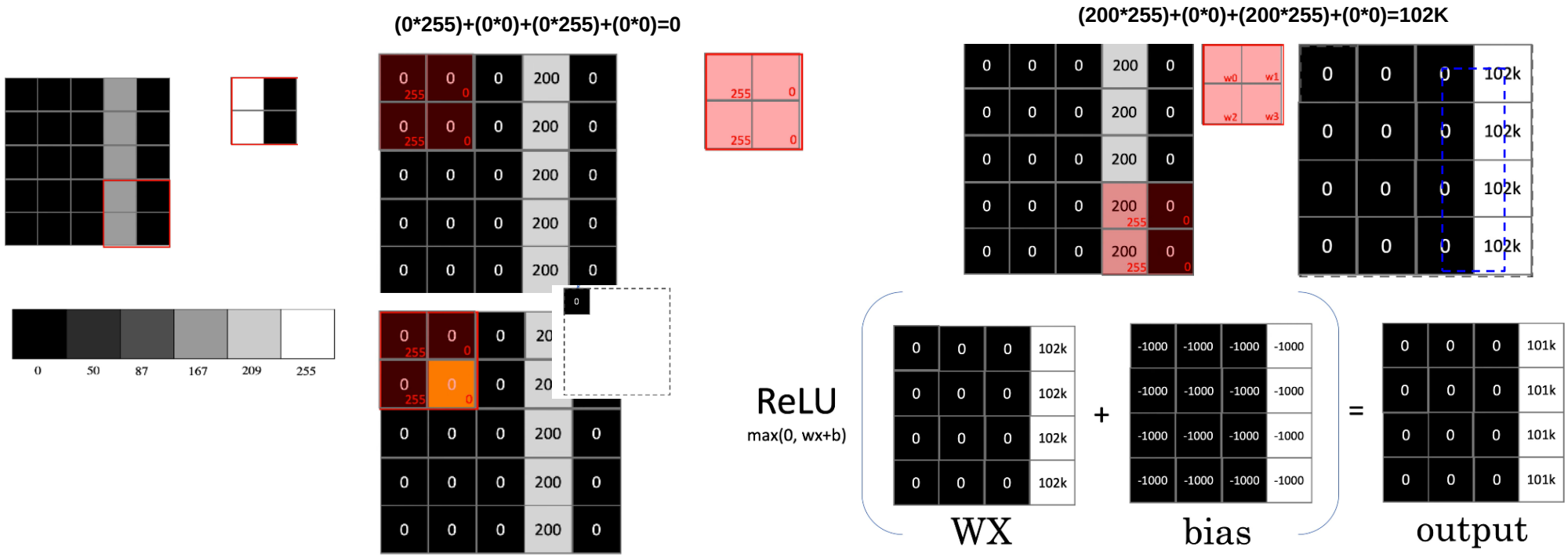
- **Filter 종류**
  - **Sobel Filter**

    : Sovel- X (vertical), Sovel-Y(holizontal)



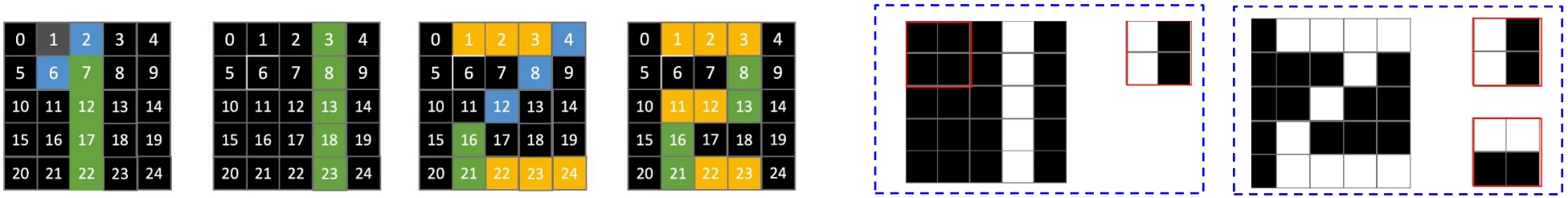
CNN – Filter

- **CNN Filter 특징**
  - 이미지 색상 인식
    : **Gray Color 은 Filter의 색상과 맞지 않아 인지할 수 없다-> Color 대신 Color 숫자 사용**
  - 인간의 눈과 같이 수직, 수평, 대각선의 색상을 인지하여 **Pixel의 Locality**를 인지할 수 있어 숫자인식이 가능
  - **kernel의 matrix of Weights는 Training 중에 Optimized 된다.**

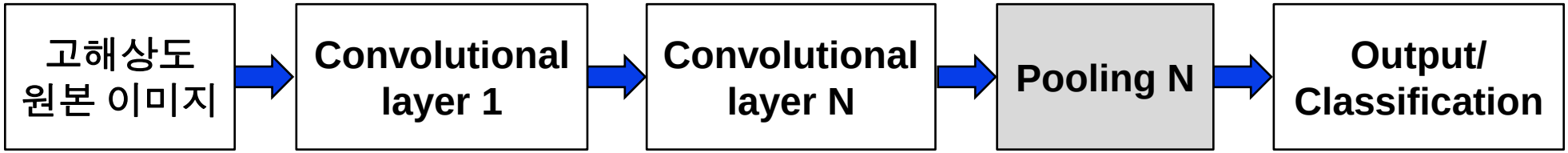


CNN – Filter

- **기존 MLP의 문제점**
  - Train Data로 Flattening후 훈련은 숫자 1 로 정확하게 할 수 있지만 Test Data는 정확도가 떨어지는 Overfitting 문제 발생
- **CNN**
  - 인간의 눈과 같이 수직의 녹색선과 수평의 노랑색 선과 파랑색의 대각선을 인지하여 Pixel의 Locality를 인지할 수 있어 숫자인식이 가능하다



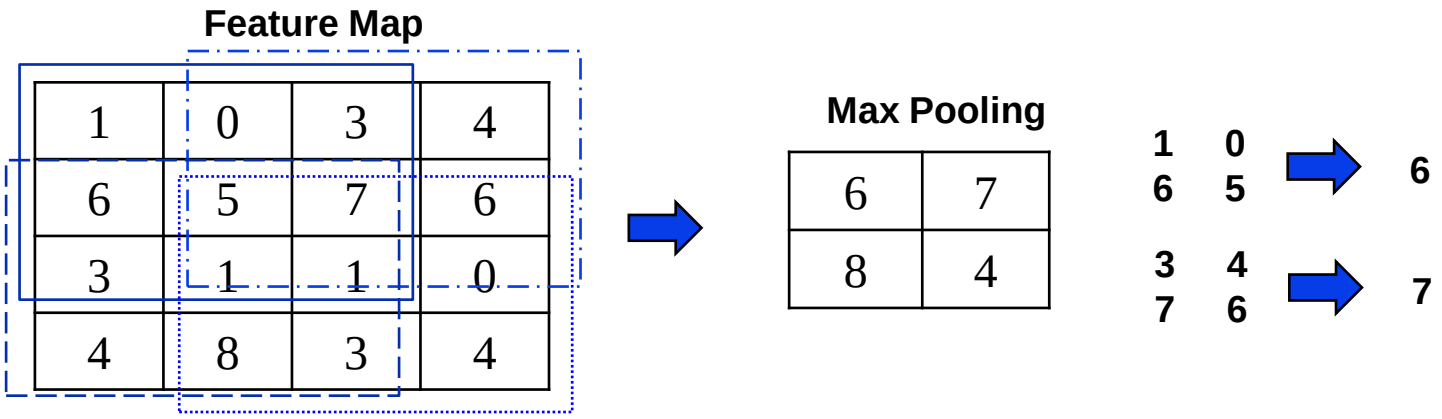
CNN – Pooling Layer



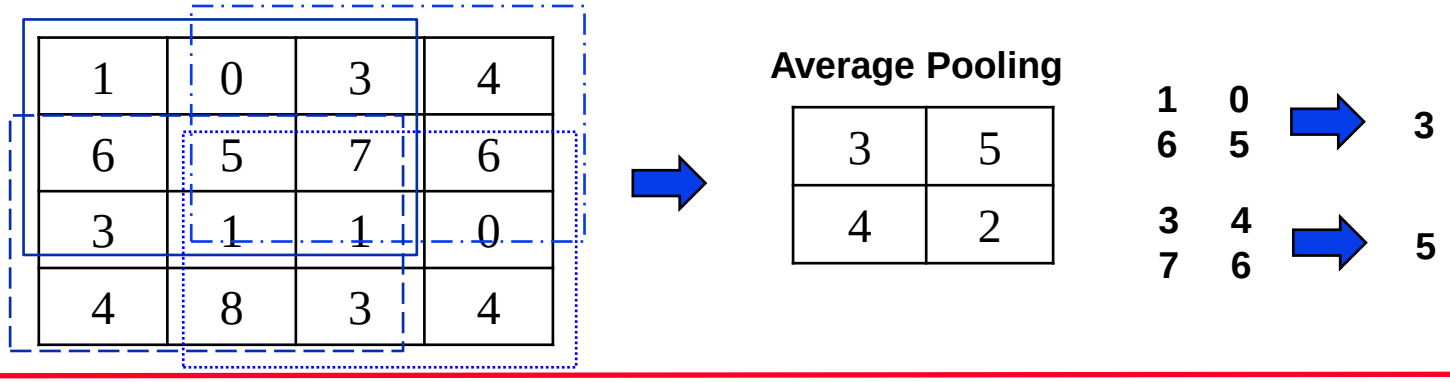
- **Activation map에 존재하는 정보 중 일부 정보를 추출(down sampling의 효과)**
- **Sub Sampling을 통해 추출된 Feature map을 인위적으로 줄이는 작업**
  - 이미지의 사이즈를 계속 유지한채로 FC layer로 가게 되면 연산량이 급속히 증가한다
  - Overfitting을 방지하며 모델의 파라메터 개수를 줄여 연산속도를 빠르게 함
- **특정 Feature, 데이터를 강조**
  : 물체의 중요한 특징을 학습할 수 있도록 해주며 CNN의 이동 불변성을 가지게 해줌
- **종류**
  - **Max Pooling, Average Pooling, Min Pooling**
    : 주로 Max Pooling 사용 -> 노이즈가 감소하고 속도가 빨라지며 영상 분별력이 향상됨
  - **주로 Pooling의 윈도우 크기와 Stride(스트라이드)의 값은 같게 설정**
    : 모든 원소가 한번씩 처리될 수 있도록 함
    예) Pooling 윈도우: 2*2 , Stride : 2
  - **Pooling 계층을 통과해도 채널수에는 변경이 없음**
  - **상황따라 생략도 가능**
    : 각 Layer의 연산 시간과 /양을 비교적 일정하게 유지하면서 시스템의 균형을 맞춤

Pooling원리 – Max Pooling, Average Pooling

- **Max Pooling**
  - 특성이 높은 부분만 추출하여 연산의 효율을 높임 (이미지 Pixel : 0~255)
  - Resizing시 해당 값 중 가장 큰값 선택



- **Average Pooling**
  - 해당값들의 평균값 사용

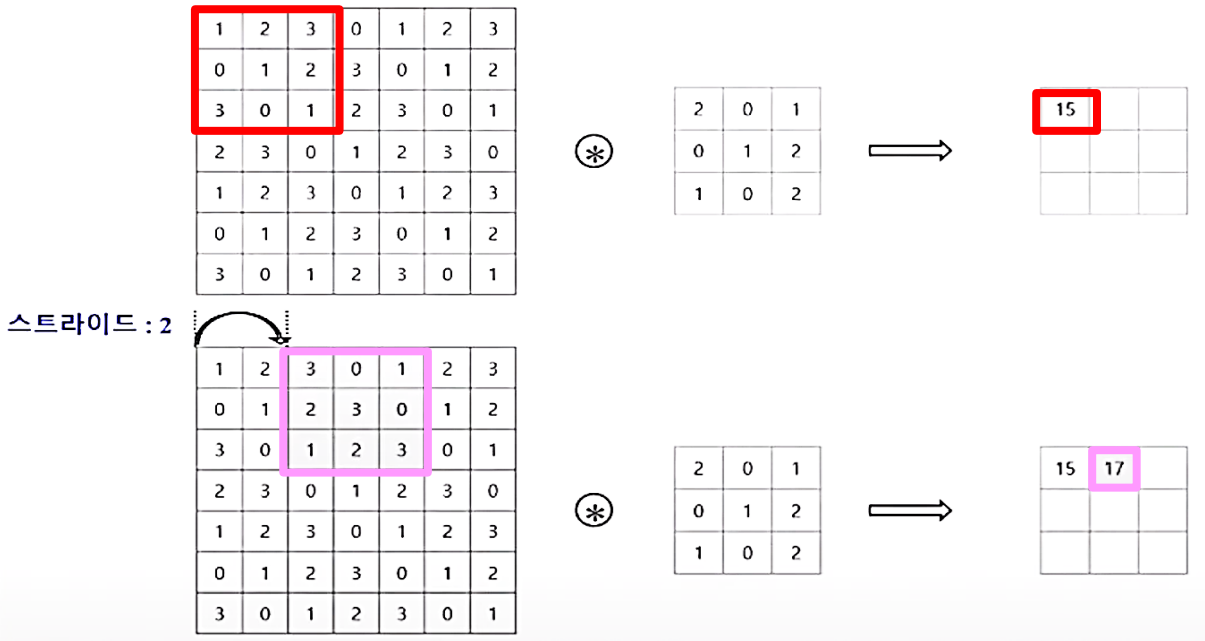


Pooling원리 – 예

- **Max Pooling**
  - 주로 Max Pooling 사용
  - Filter 크기가 크면 많이 뭉개짐
  - 0->255 갈수록 밝아짐

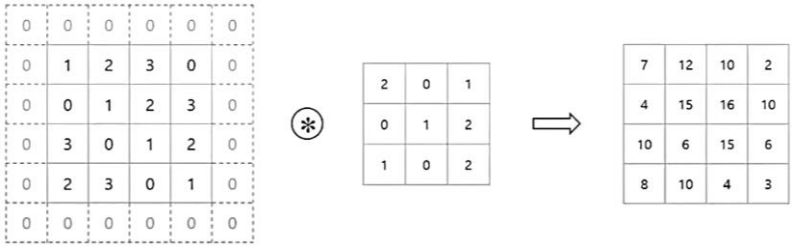
CNN- Stride

- **Convolution Filter의 Sliding Window가 이동하는 간격, 필터를 적용하는 위치의 간격**
  - Stride가 커지면 출력의 크기가 작아짐, 정보손실 가능성 (Hyper parameter)
  - Stride를 키우면 불필요한 특성을 제거하고 연산속도를 향상시킬 수 있음
    : 연산시 데이터를 중첩시키지 않고 효율성을 높이기 위해 사용
  - 중요 Feature의 손실을 반드시 의미하지는 않음
    참조) stride=2 , kernel_size =2를 통해 특징맵의 사이즈를 줄이는 역할
  - stride 후에 남는 열은 버리게 된다. -> Padding 사용



CNN- Padding

- **필터 사이즈와 함께 입력 이미지와 출력 이미지의 사이즈를 결정하기 위해 사용**
  - Feature map의 출력 데이터 크기는 Convolution Layer을 지날 때마다 사이즈가 축소
  : 합성곱 연산 후 데이터 사이즈가 (n- f+1) * (n- f+1) Pixel Size로 축소됨
  - **Stride 후에 가장 자리의 정보가 사라지는 문제 방지**
- **출력 크기 조정으로 입력 데이터와 출력 데이터의 크기를 맞춤**
  - **입력 데이터의 주변을 특정 값으로 채우는 기법**
  - Zero padding :주로 0을 사용
  - Padding을 한 후 대칭을 이루기 위해 Filter의 사이즈를 주로 홀수로 정함
  - 가장자리의 연산 횟수를 증가시켜 Edge Feature들의 특징을 강화
- Padding – valid
  - 패딩을 주지 않음
  - Padding =0 (0으로 채워진 테두리가 아니라 패딩을 주지 않는다는 의미)
- Padding – same
  - 패딩을 주어 입력 이미지의 크기와 연산후의 이미지 크기를 같게 함
- **만약, 필터의 크기가 k이면**
  - 패딩의 크기는 $p = \frac{k-1}{2}$ (단, Stride =1)



(4*4) **(3*3)** **(4*4)**

입력 데이터 (제로패딩:1)->(6*6) **필터** **출력데이터**

CNN- 출력의 크기

- 출력데이터의 크기

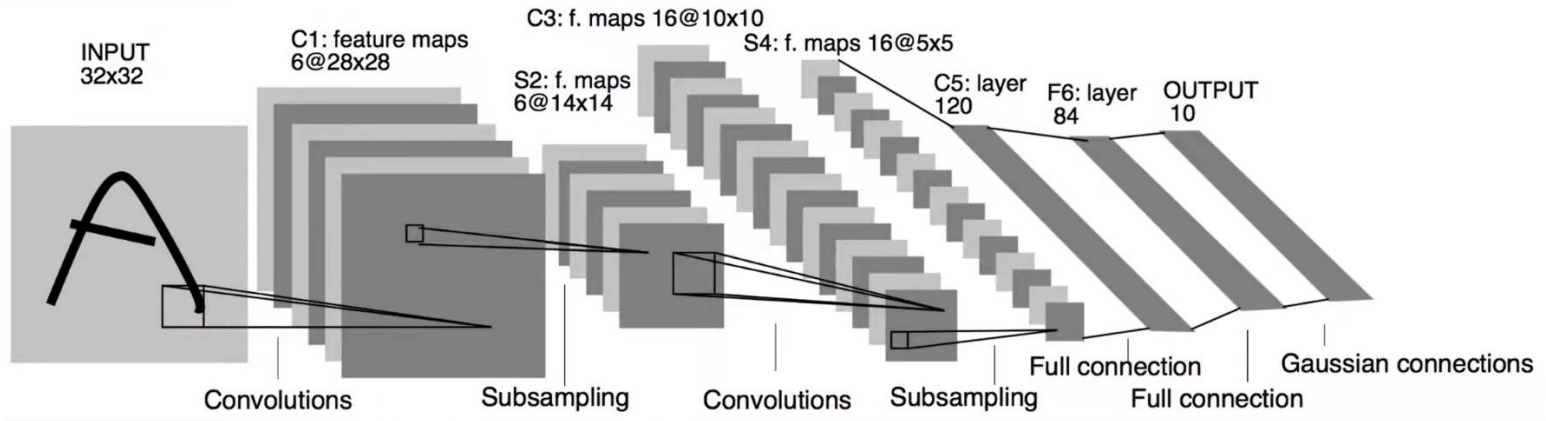
$$OH = \frac{H+2P-FH}{S} + 1$$

$$OW = \frac{W+2P-FW}{S} + 1$$

- 입력 크기 : (H,W) , 출력크기: (OH,OW)
- 필터크기 : (FH,FW)
- 패딩, 스트라이드 : P, S

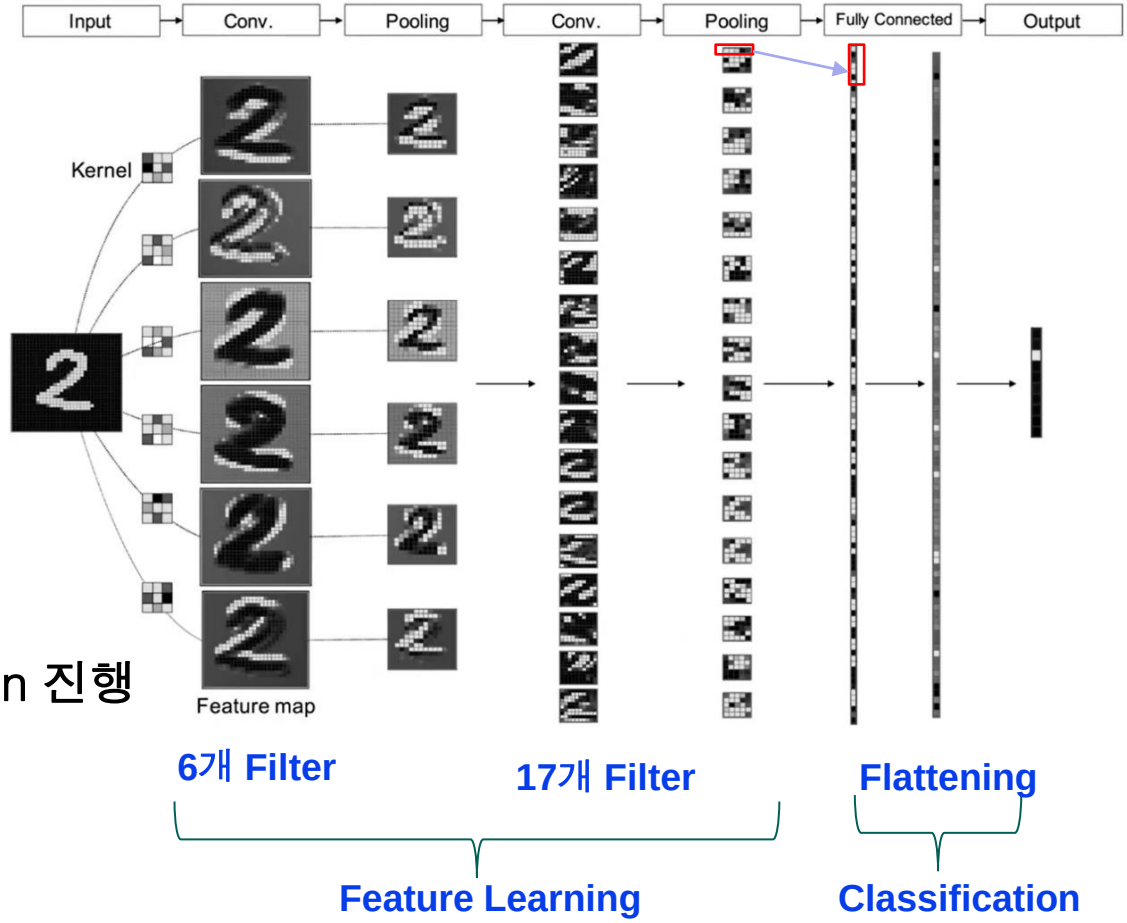
CNN- 완전 연결층(Fully Connected Layer)

- **완전 연결계층(Fully Connected Layer)**
  - 추출된 특징값을 기존 Neural network에 넣어 결과를 도출
- **Flattening**
  : Fully Connected Layer은 3차원에서 1차원 데이터로 평탄화된 행렬로 이미지 분류
  - Drop Out 기법으로 성능향상과 Overfitting 방지
  - 활성화 함수: ReLU, 이미지 분류 : Softmax 등 사용
- **Convolution Layer와 완전 연결 계층(Fully-Connected Layer)과의 차이**
  - 완전연결계층은 이미지와 같은 데이터의 형상(3차원)을 무시됨
    :모든 입력 데이터를 동등하게 취급 -> 데이터의 특징을 잃어버리게 됨
  - **Convolution Layer는 이미지 픽셀 사이의 관계를 고려 <u>공간정보를 유지</u>**
    :이미지가 왜곡되더라도 <u>이미지의 부분적 특성을 추출할 수 있어 올바른 성능 도출</u>

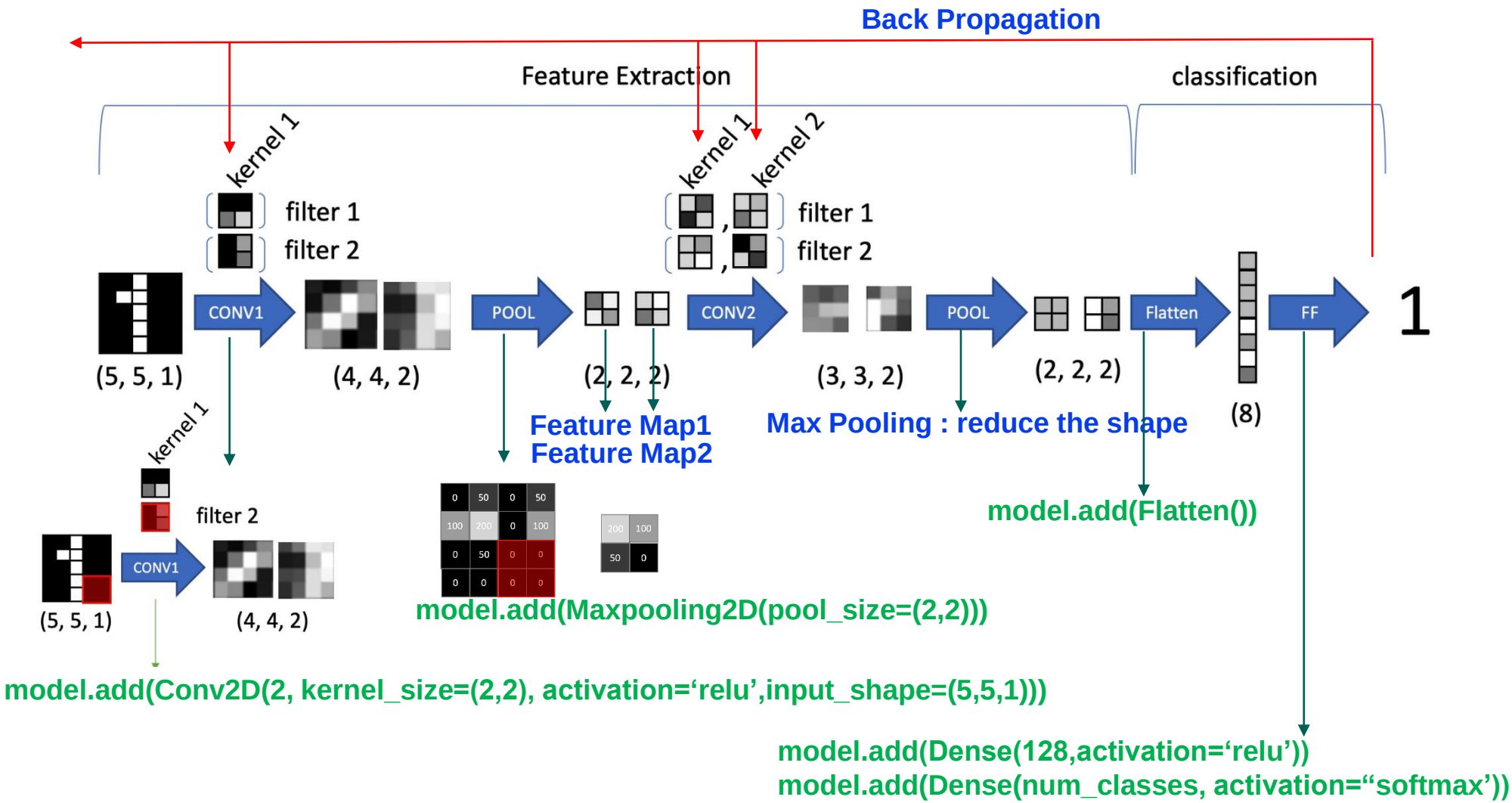


CNN - Overview

1. 6개의 필터로 컨볼루션 수행
   – 6개의 feature map 도출
2. Max Pooling으로 사이즈 축소
3. 다음층에서 17개 필터로 convolution과 pooling 수행
   :상위계층으로 올라갈 수로 인간은 인지 어려움
4. Flattening 수행으로 완전 연결층 입력 데이터 구성
   : 3 차원 –> 1차원
5. ReLU 활성화 함수
6. 완전 연결층에서 NN수행후 Classification 진행
   – Softmax: 다중 분류 수행



CNN – Flow


Back Propagation
Feature Extraction
classification
kernel 1
filter 1
filter 2
CONV1
(5, 5, 1)
(4, 4, 2)
POOL
kernel 1
kernel 2
filter 1
filter 2
(2, 2, 2)
CONV2
(3, 3, 2)
POOL
(2, 2, 2)
Flatten
(8)
FF
1
Feature Map1
Feature Map2
Max Pooling : reduce the shape
model.add(Flatten())
model.add(Maxpooling2D(pool_size=(2,2)))
model.add(Conv2D(2, kernel_size=(2,2), activation='relu',input_shape=(5,5,1)))
model.add(Dense(128,activation='relu'))
model.add(Dense(num_classes, activation="softmax'))