CNN(Convolution Neural Network)

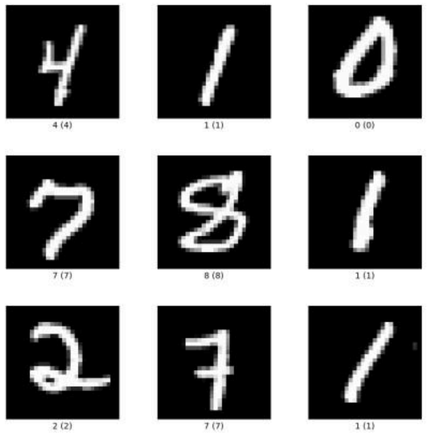
홍익대 소트프웨어융합학과 c189044 이민욱

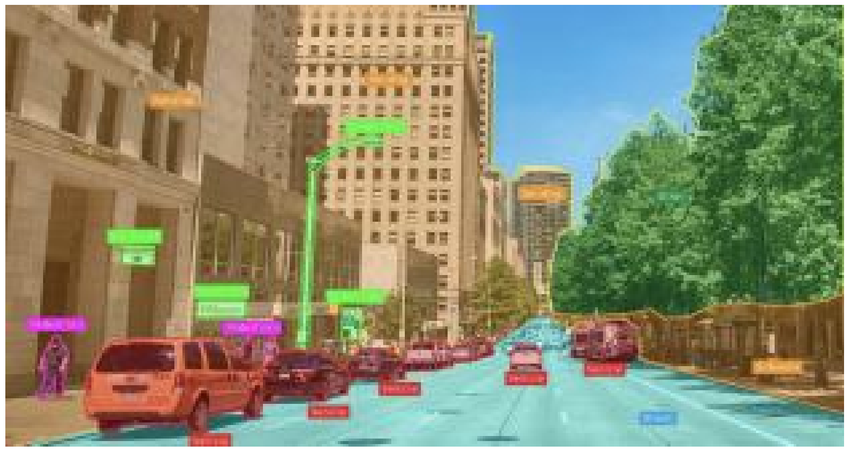
Background for CNN

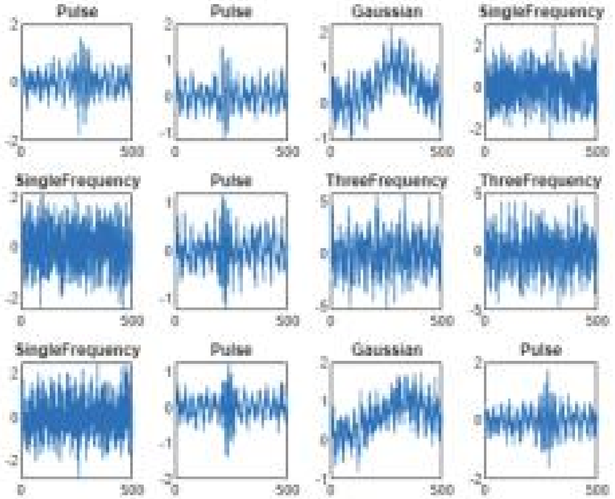
- Image Classification
- Representation Learing
- Dimension Reduction
- Traditional Machine Learnig vs Deep Learing

Background for CNN

- **Computer Vision : 인간의 시각과 관련된 능력을 컴퓨터에 학습시키는 분야.**
- **주로 이미지, 비디오 등의 비정형 데이터를 다룸.**

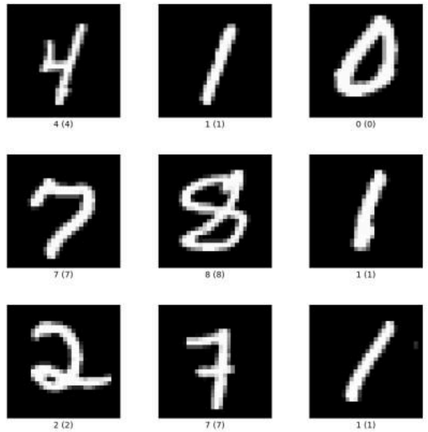






Representation Learing

- Image Classification ex) 강아지와 고양이, MNIST의 숫자 데이터 분류
- feature(특징): Sample을 잘 특정짓는 특징.



Traditional Machine Learing vs Deep Learing

< Traditional Machine Learing >

- 사람이 데이터 분석 후 가정.
- 가정에 따라 전처리를 하여 feature 추출
- 추출된 feature를 model에 넣어 학습.

< Deep Learning >

- Raw Data를 최소한의 전처리를 수행.
- hidden layer가 자동 feature 추출
- end to end

Feature Vector

Feature vector

- 각 특징들을 모아서 하나의 vector로 만든 것
- 각 차원(dimension)은 어떤 속성에 대한 level을 나타냄.
- Feature Vector를 통해 샘플 사이의 거리(유사도)를 계산할 수 있다.

	키	몸무게	나이	월 소득	집 평수	차 배기량	월 지출
홍길동	175	70	35	432	24	1600	345

Continuous Value vs Categorical Value

- **Continuous Value: 비슷한 값은 비슷한 의미를 가진다. ex) 키, 몸무게**

	키	몸무게	나이	월 소득	집 평수	차 배기량	월 지출
홍길동	175	70	35	432	24	1600	345

- **Categorical Value: 비슷한 값이라도 상관없는 의미.**

단어를 사전 순으로 index에 mapping.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
가위	공책	교과서	노트	딱풀	볼펜	색연필	샤프	싸인펜	연필	자	지우개	책상	칼	필기장	필통

One-hot encoding

- 크기가 의미를 갖는 integer 값 대신,
- 한 개의 1과 n-1개의 0으로 이루어진 n차원의 벡터.
- Sparse Vector （↔ Dense Vector)
- 서로 다른 두 벡터가 항상 직교.
- 어떤 범주 간에도 유사도가 없다.
- 하지만 의미적 유사도가 필요하면 Dense Vector로 변환 (Embedding)

	idx	가위	공책	교과서	노트	딱풀	볼펜	색연필	샤프	싸인펜	연필	자	지우개	책상	칼	필기장	필통
공책	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
노트	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
지우개	11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

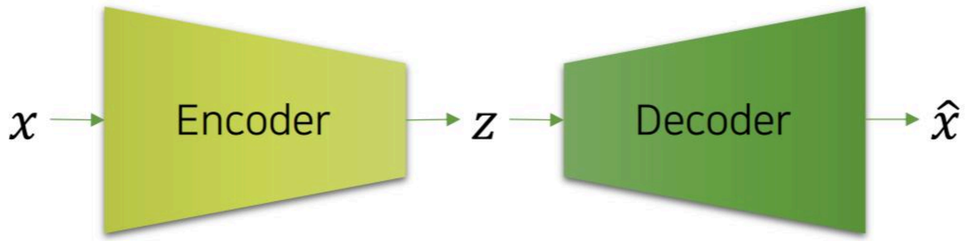
n개의 항목 → n차원

*임베딩(embedding): 어떤 이산적(discrete)이고 복잡한 데이터를, 의미 있는 연속적(dense) 벡터 공간에 매핑하는 것.

Autoencoder

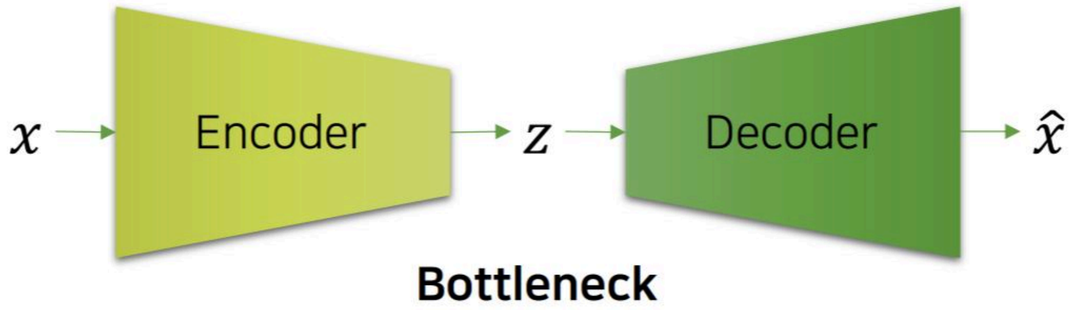
- Encoder와 Decoder를 통해 압축과 해제를 실행
  - encoder는 입력(x)의 정보를 최대한 보존하도록 손실 압축을 진행
  - decoder는 중간 결과물(z)를 입력(x)과 같아지도록 압축 해제를 진행

- 복원을 성공적으로 하기 위해,
- Autoencoder는 feature을 추출하는 방법을 자동으로 학습



Bottleneck

- 입력(x)에 비해 작은 차원으로 구성
- 정보의 선택과 압축이 발생, 차원에 따라 압축의 정도를 결정.
- z는 입력(x)에 대한 feature vector라고 할 수 있다.
  - 압축의 효율이 높아야 하므로, 입력에 비해 dense vector
- Encoder에 통과시키는 것은 feature vector에 대한 embedding 과정

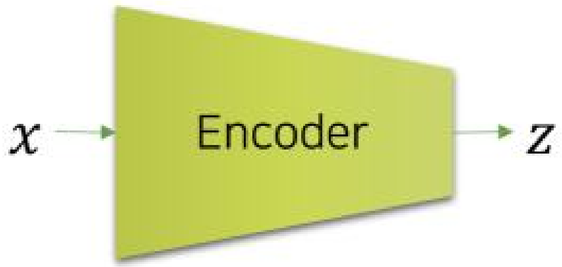


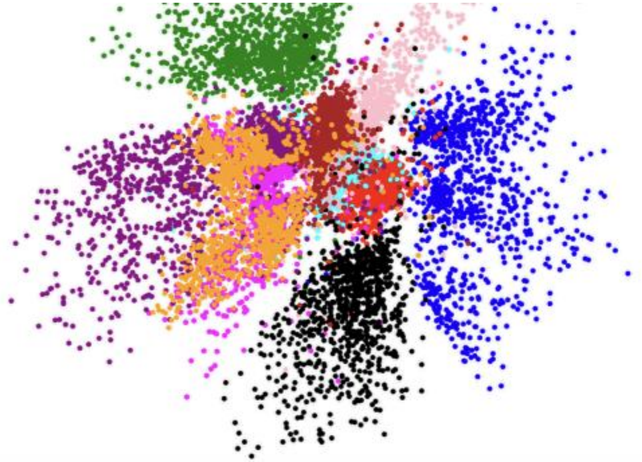
*임베딩(embedding): 어떤 이산적(discrete)이고 복잡한 데이터를, 의미 있는 연속적(dense) 벡터 공간에 매핑하는 것.

Hidden Space

- 인코더의 결과물 z를 plot 하였을 때, 비슷한 샘플들은 유사도를 가진다.
  - z는 feature vector이며, 이는 의미있는 연속적인 dense vector로 embedding 된 결과

- plot이 나타내는 공간이 hidden space

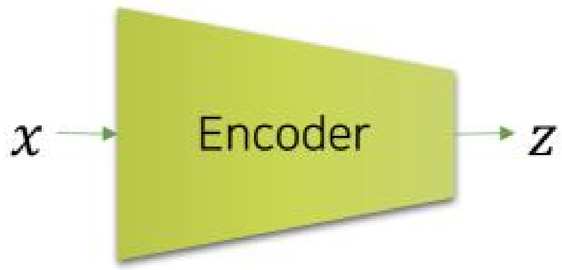


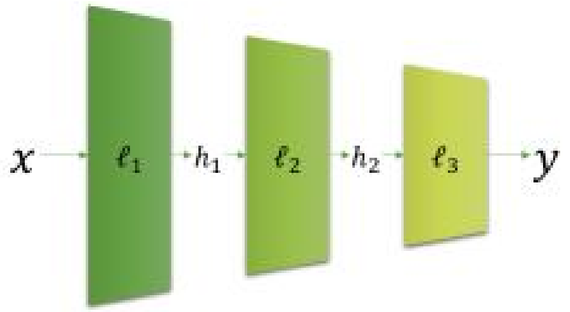


hidden vector

- Encoder의 결과물은 hidden vetor
- 확장하여 모든 DNN layer의 결과물을 hidden vector라고 할 수 있다
  - Autoencoder에서 x_hat으로 잘 복원되기 위한 feature vector
  - DNN에서 입력 데이터을 더 잘 예측하기 위한 feature vector

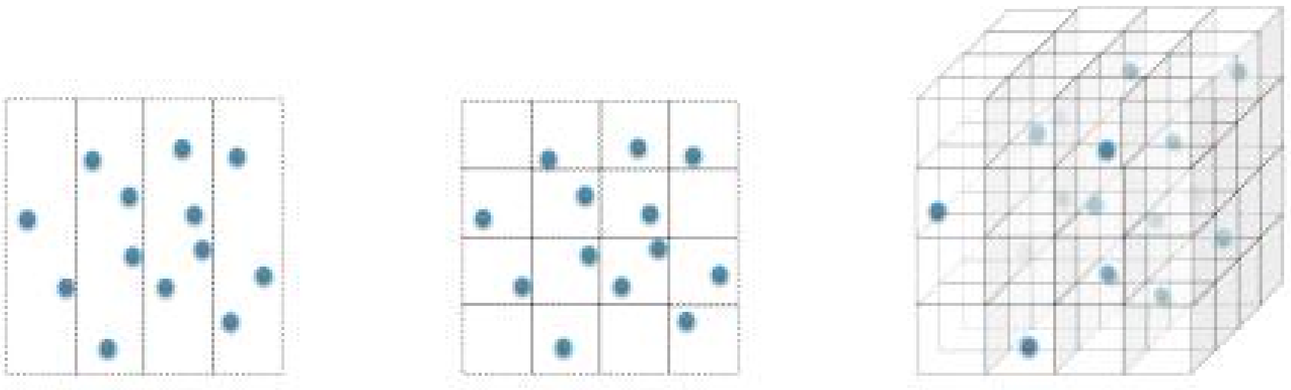
- 신경망(layer)을 통과시키는 것은
- input space에서 output space로의 mapping 과정
- 고차원 공간 → 저차원 공간





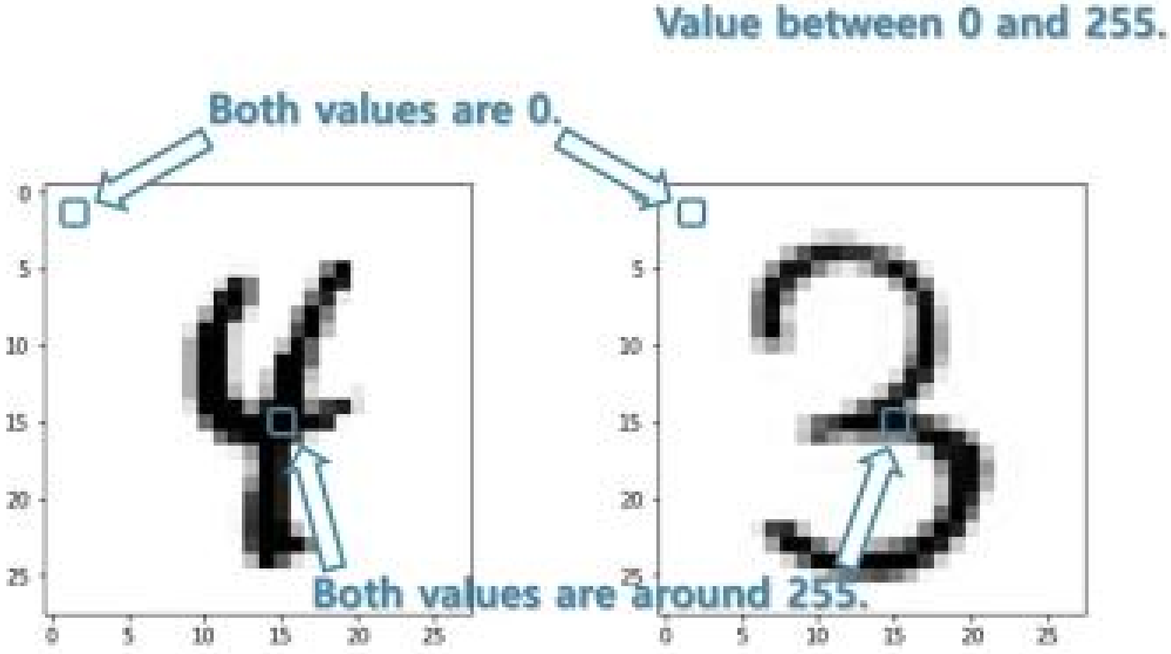
Curse of Dimensionality

- 차원이 높을수록 데이터는 희소하게(sparse)하게 분포되어 학습이 어려워 진다.
- 같은 정보의 데이터를 표현할 때, 차원이 높아질수록 희소성이 증가
- 데이터의 좋은 Feature을 위해 낮은 차원에서 표현해야 함.
  - 하지만 너무 낮은 차원은 feature을 충분히 표현하기 어려움.



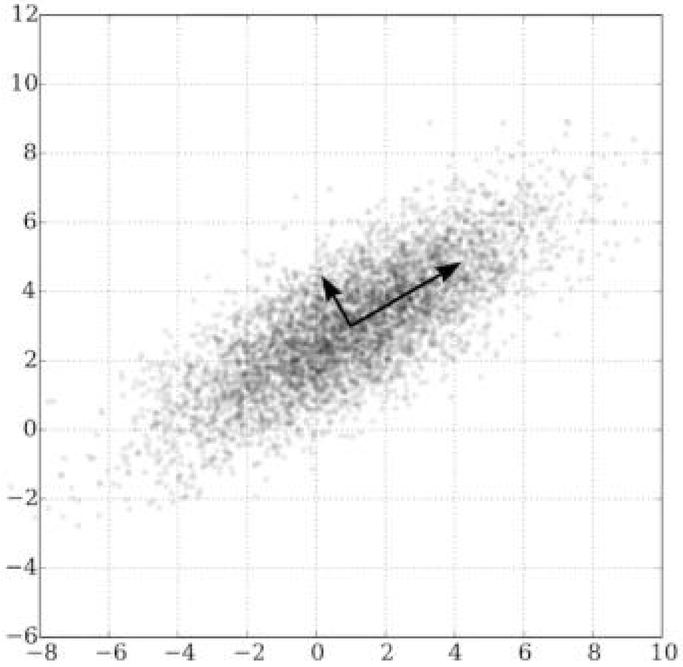
Example : MNIST

- MNIST = 28 × 28 = 784차원
- 하나의 샘플은 784차원이고, 이는 희소(sprase)하게 분포되어 있다
- 즉, 적당한 차원으로 축소하는 것이 유리하다



Linear Dimension Reduction: PCA

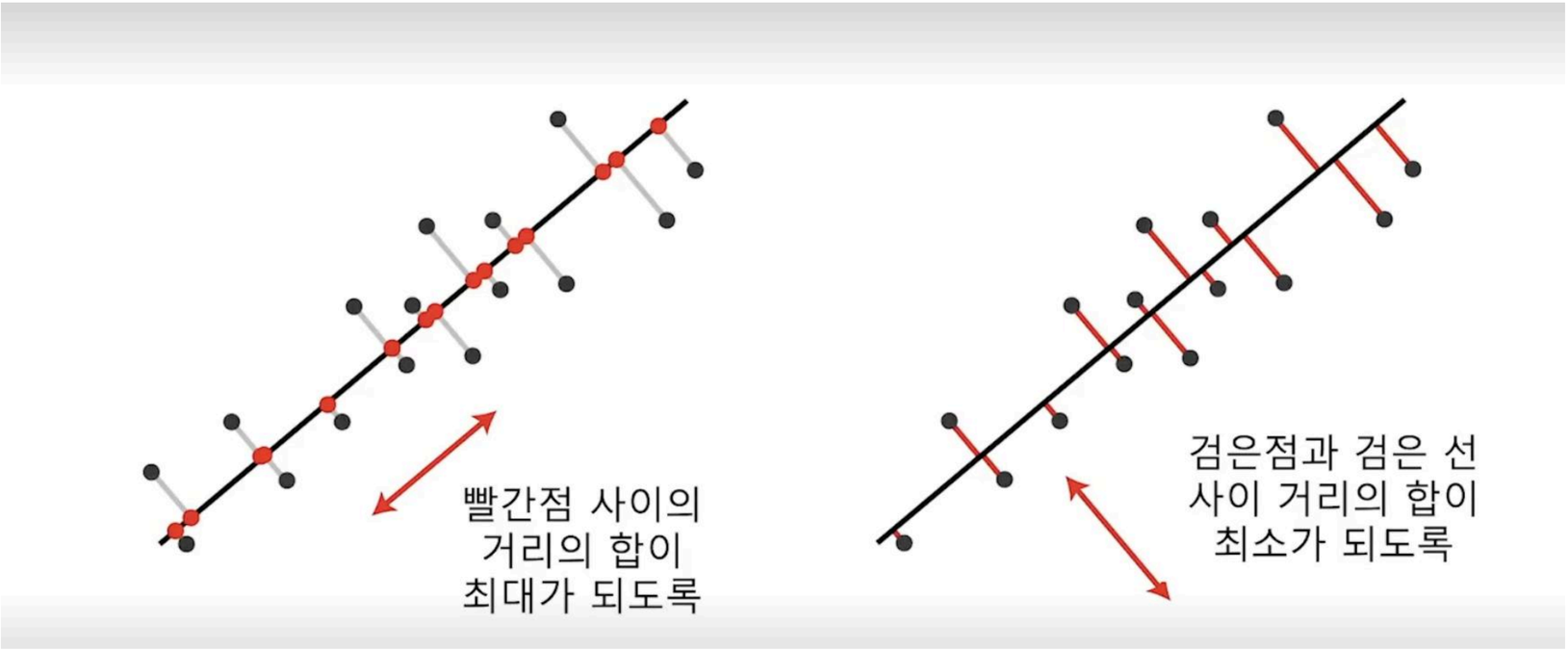
- n차원의 공간에 샘플들의 분포가 주어져 있을 때, 분포를 잘 설명하기 위한 새로운 axis을 찾는 과정



<2차원 공간의 샘플 분포>

Linear Dimension Reduction: PCA

- **새로운 axis는 두 가지 조건을 만족해야 한다.**
- **projection**

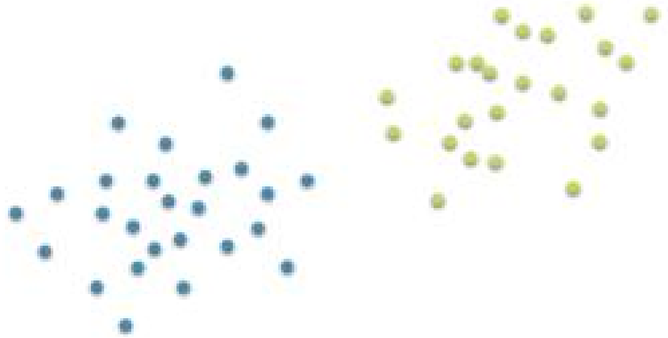


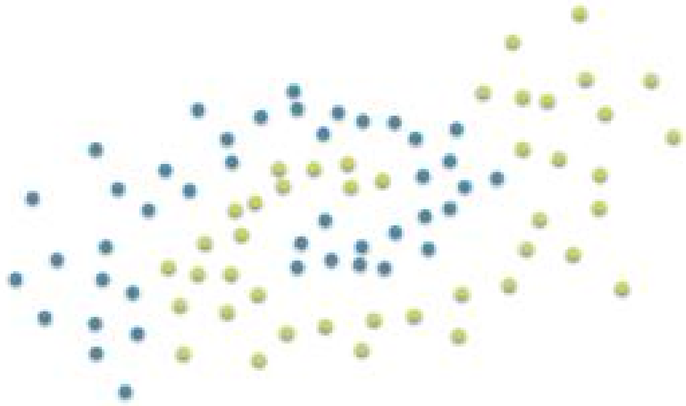
Linear Dimension Reduction: PCA

- 새로운 axis에 샘플을 projection하면 차원 축소가 가능하다

Limitation of Linear Dimension Reduction

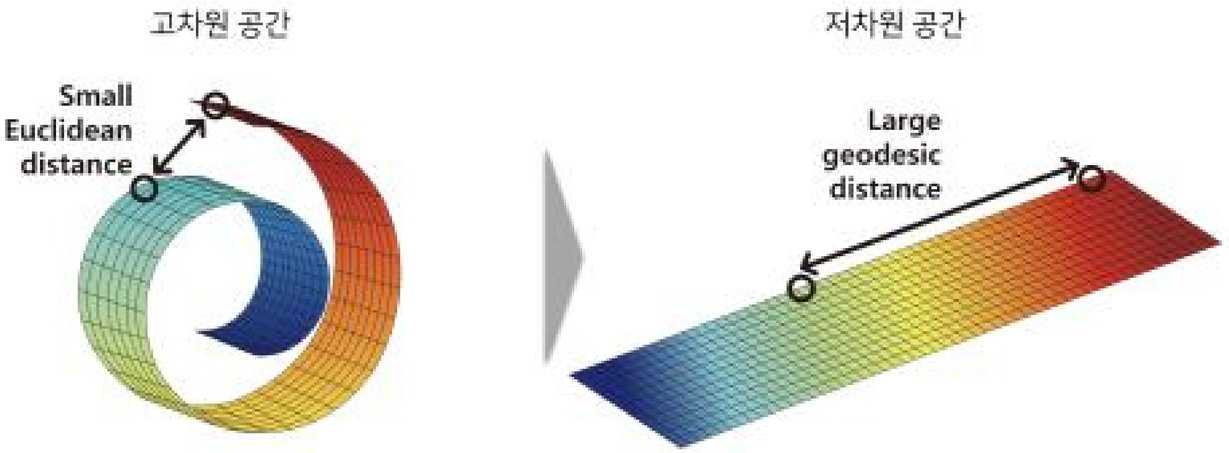
- Binary Classification in 2D
- Non Linear dicision boundary
- Deep Learing,
- non linear 차원 축소 과정을 통해 압축을 배운다.





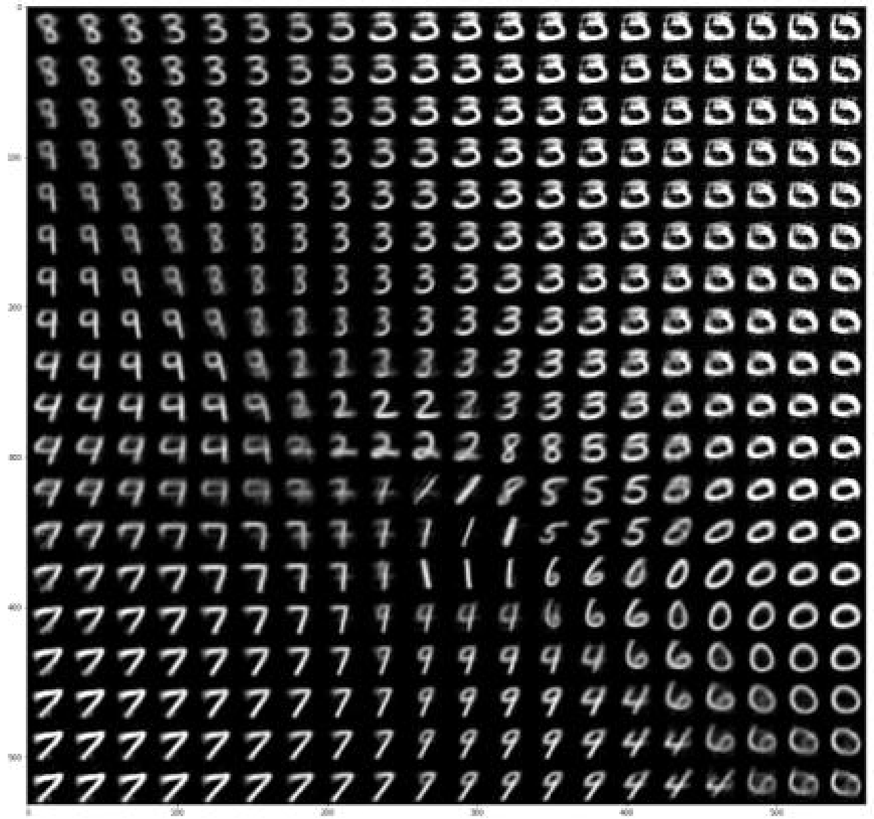
Manifold Hypothesis

- 고차원 공간의 샘플들이 저차원 다양체(manifold)의 형태로 분포되어 있다는 가정
  - 따라서 다양체를 해당 차원의 공간에 mapping할 수 있다
- 고차원 공간 상의 거리와 저차원 공간 상의 거리가 다르다

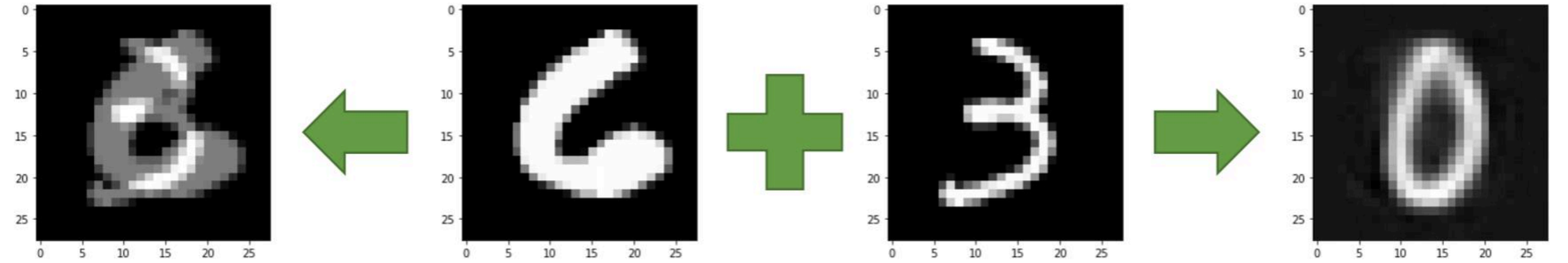


MNIST Dimension Reduction

- MNIST : 28 × 28 = 784차원
  - 786차원의 샘플 → 2D Space mapping

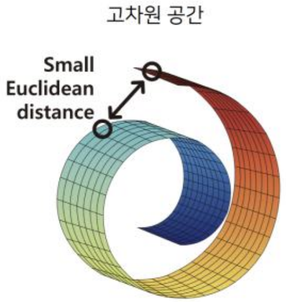


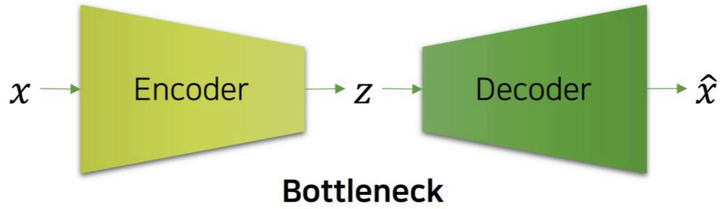
MNIST Dimension Reduction

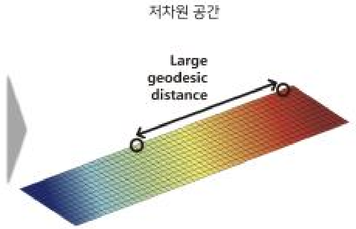


High Dimensional Space

Lower Dimensional Space



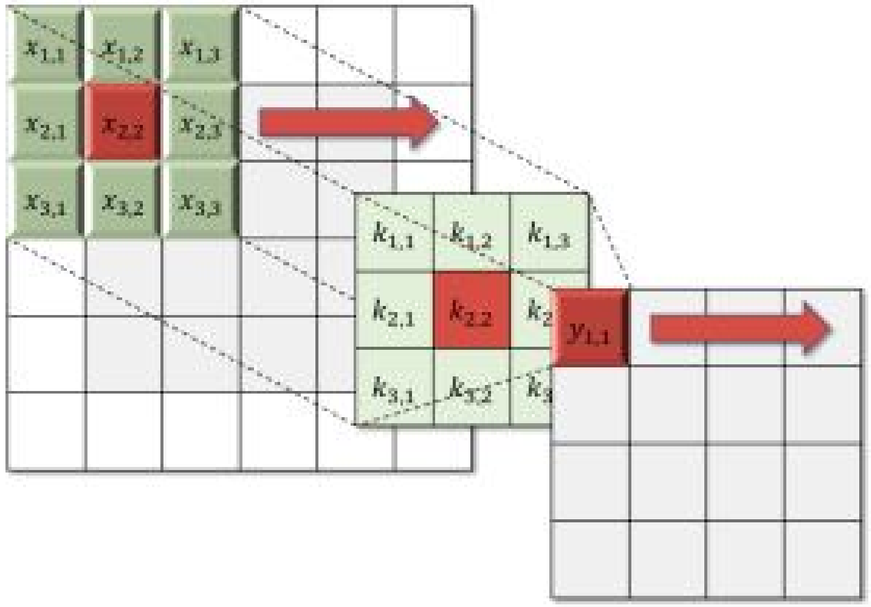




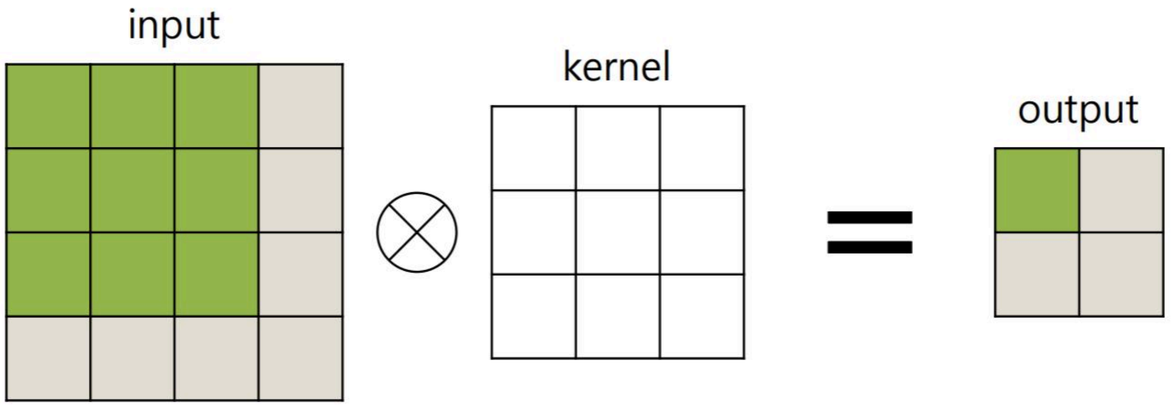
CNN for Image Classification

Convolution Operation

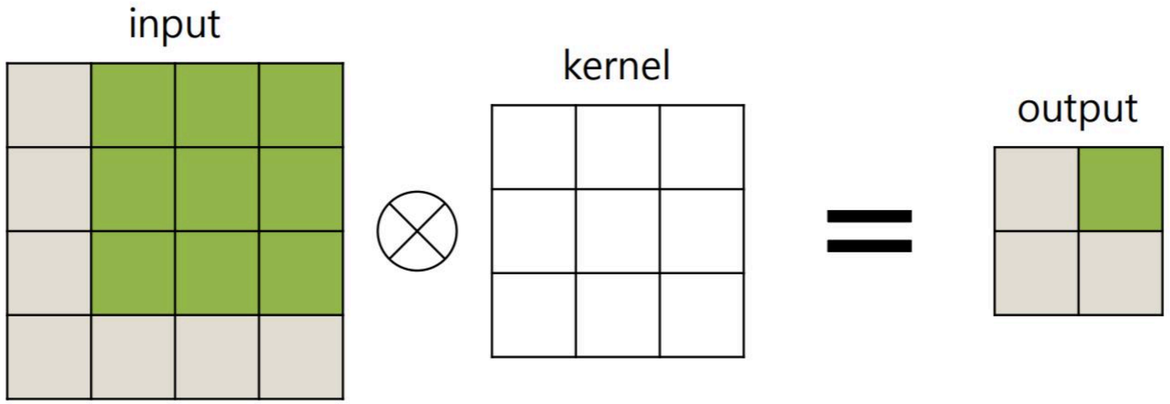
convolution filter(kernel)가 입력(x)에 대하여 돌아가며 동작.



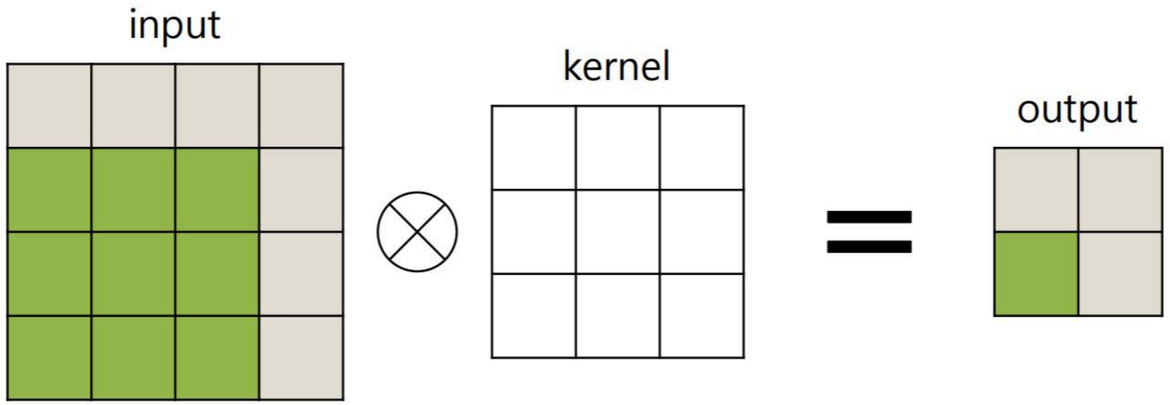
Convolutional Neural Network



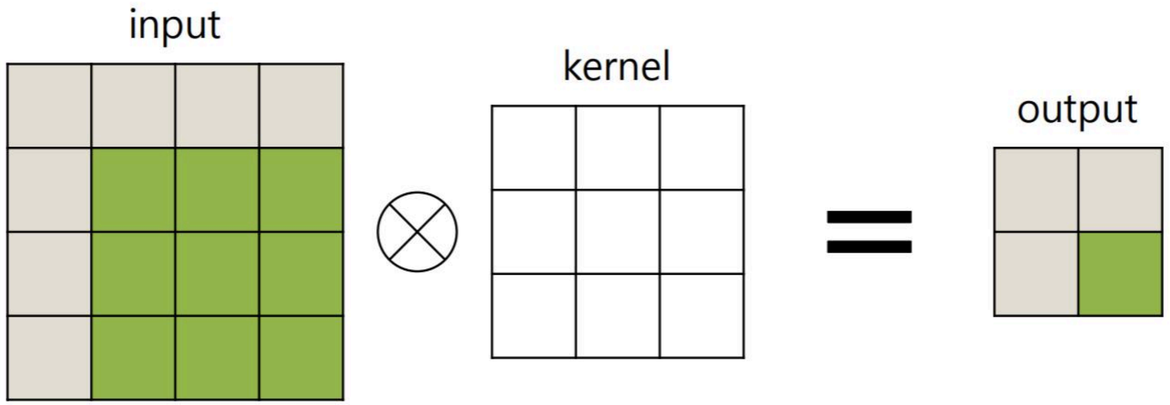
Convolutional Neural Network



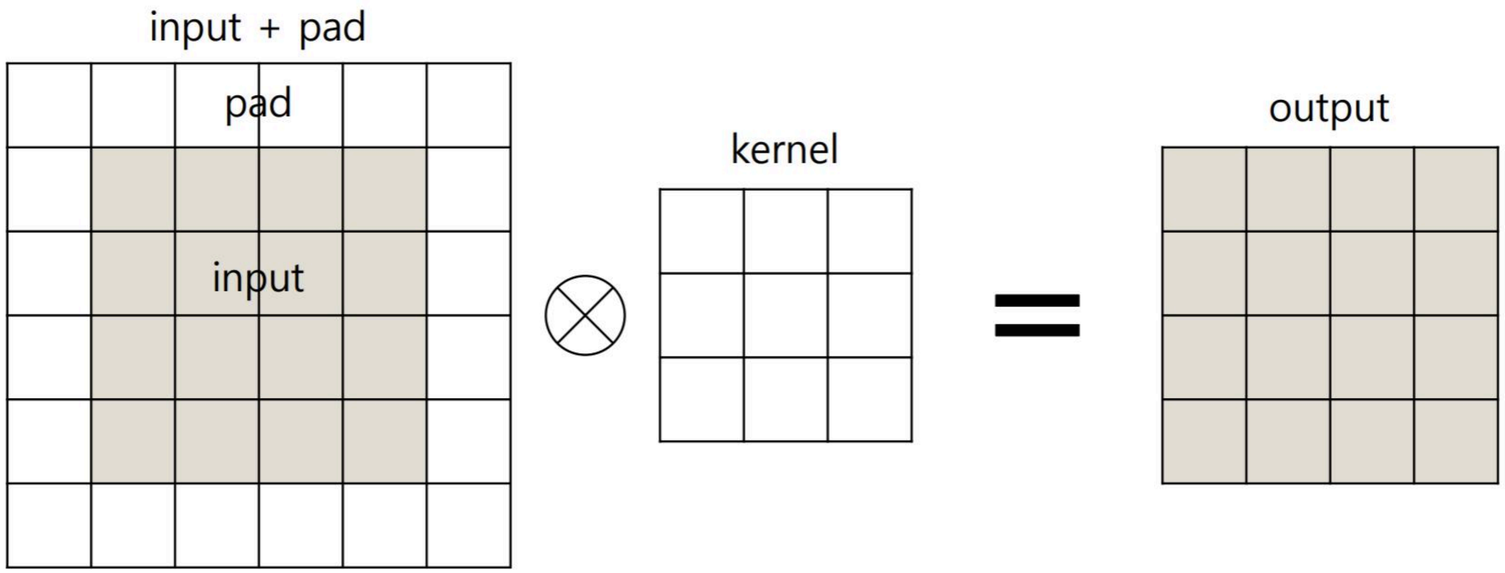
Convolutional Neural Network



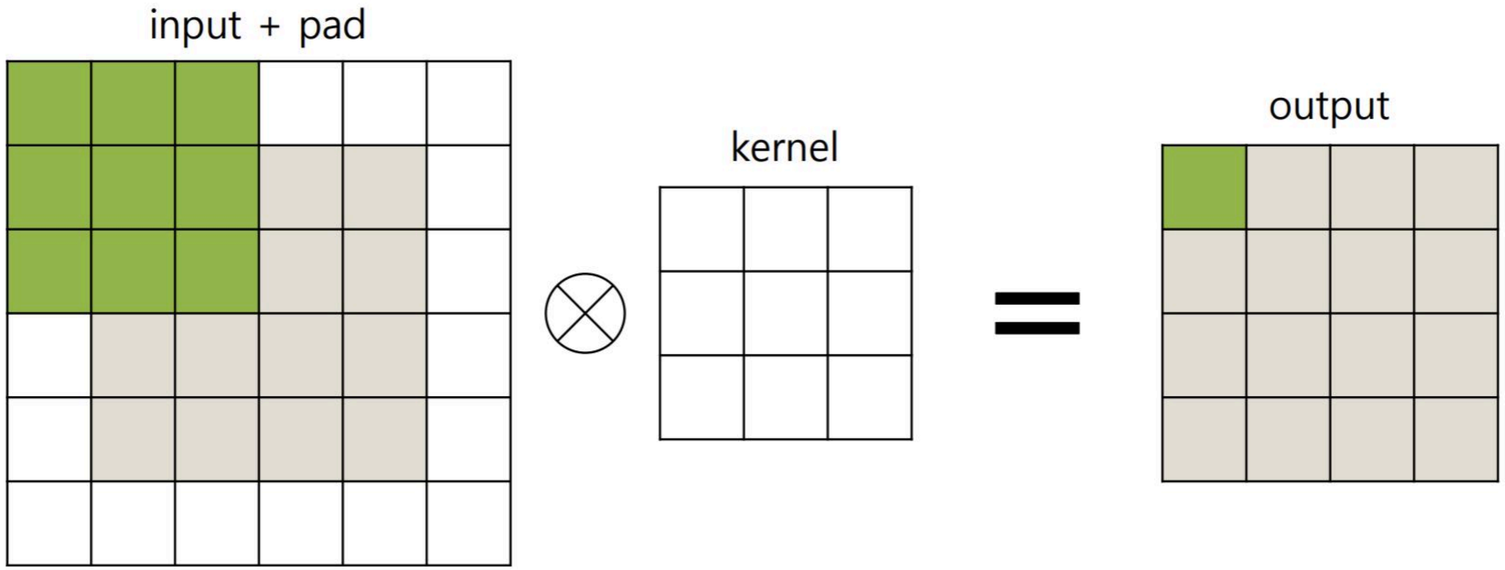
Convolutional Neural Network



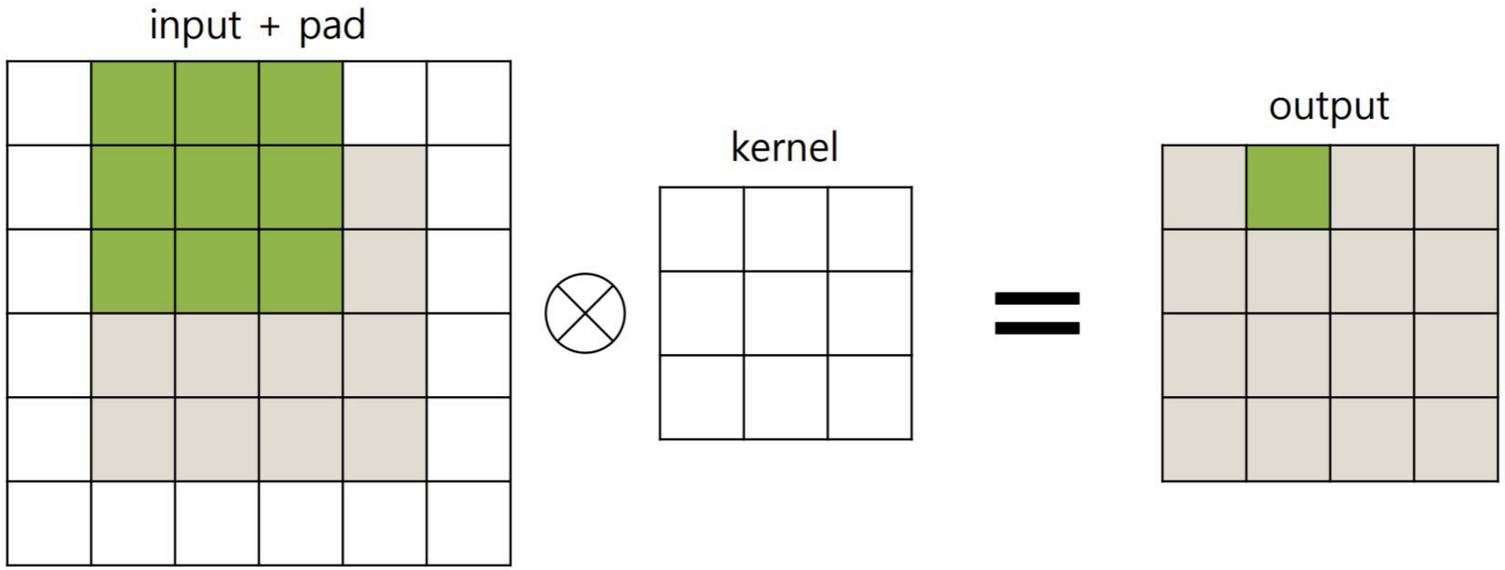
Convolutional Neural Network (with Pad)



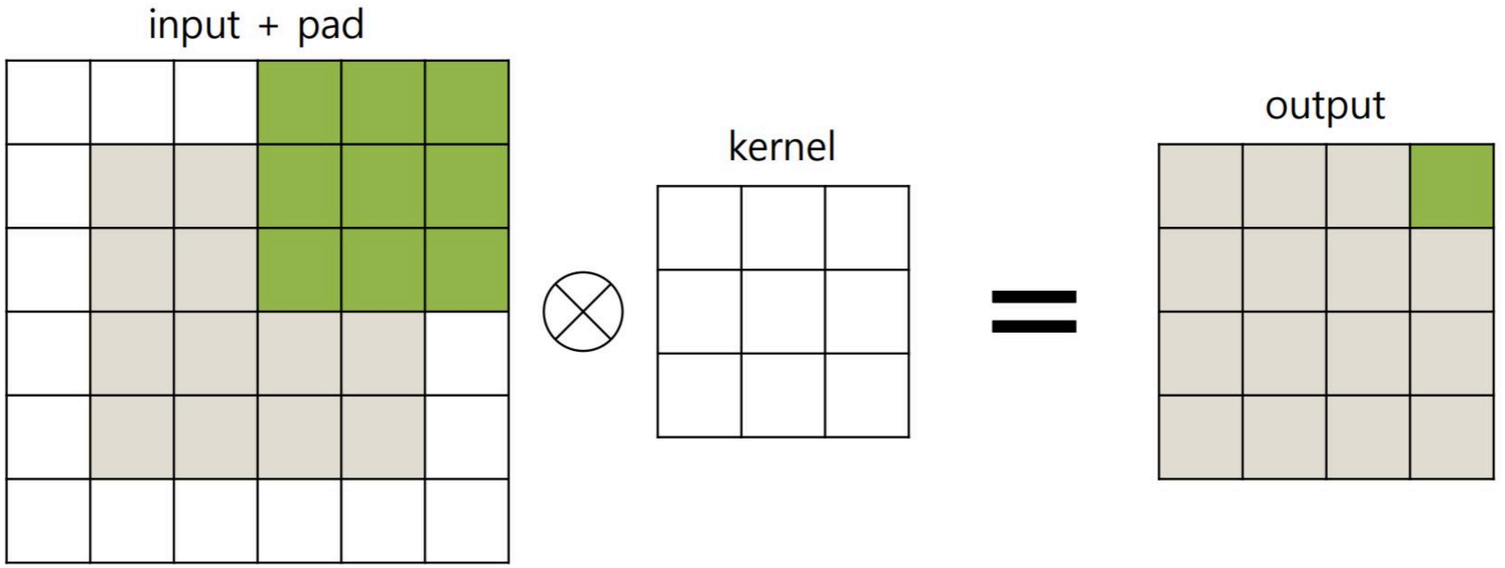
Convolutional Neural Network (with Pad)



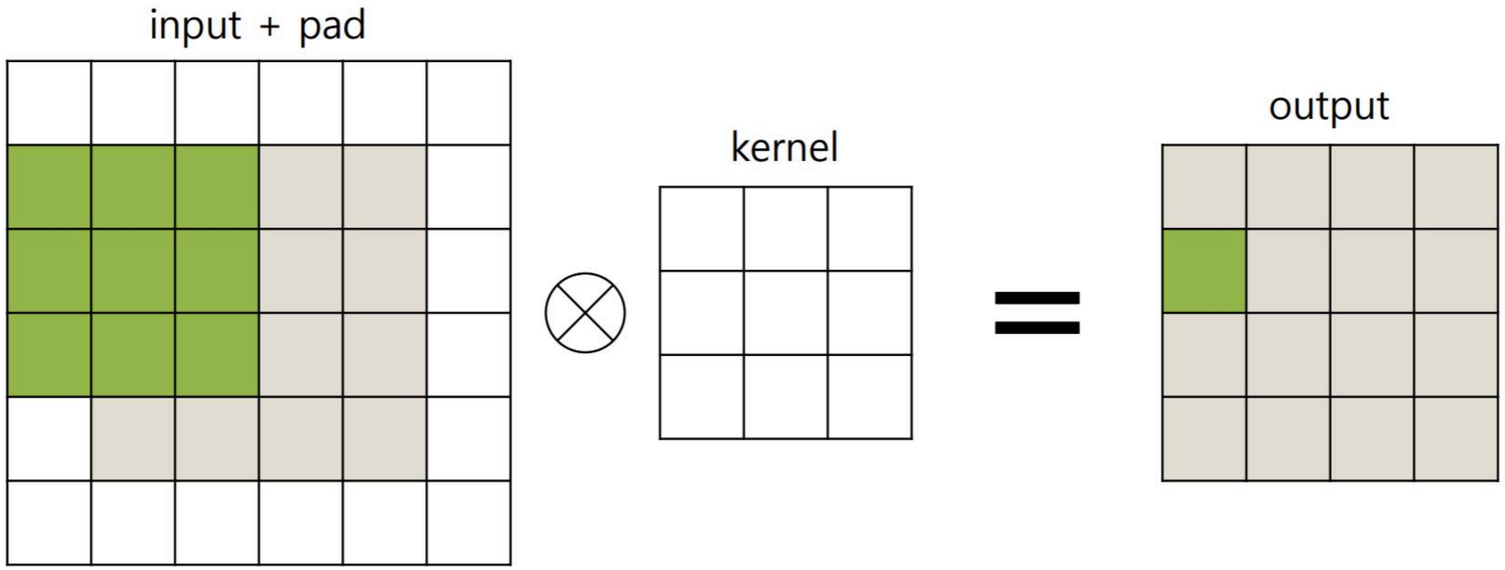
Convolutional Neural Network (with Pad)



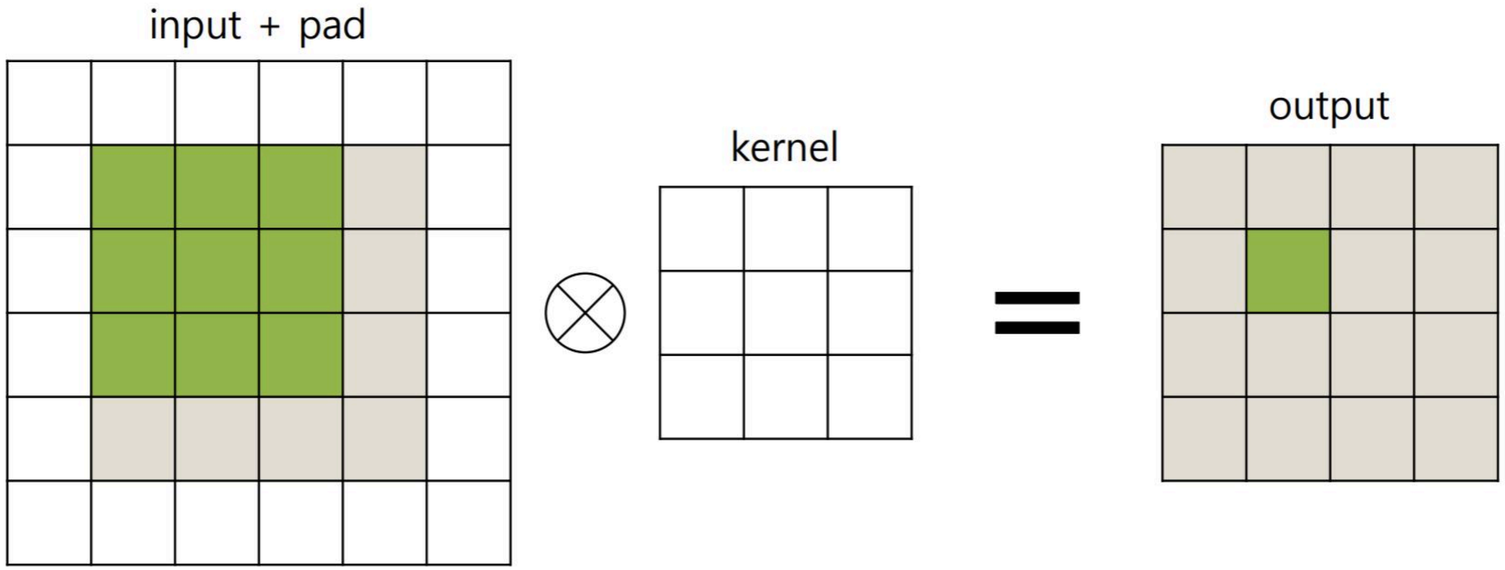
Convolutional Neural Network (with Pad)



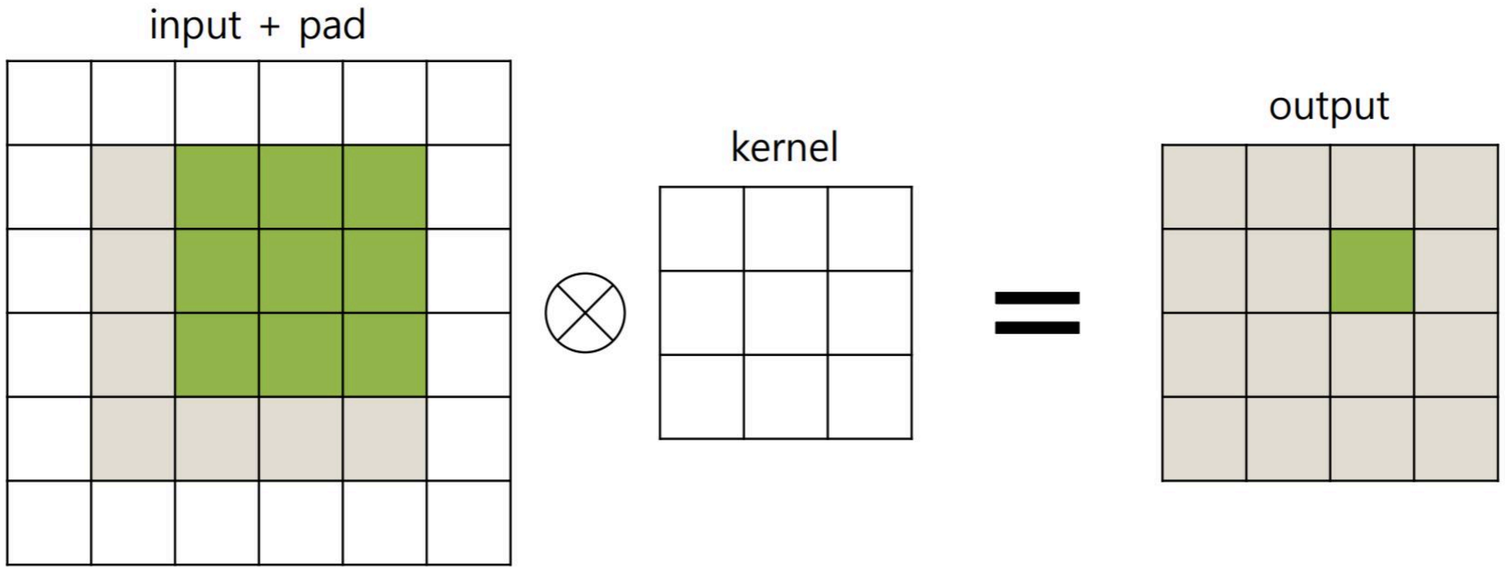
Convolutional Neural Network (with Pad)



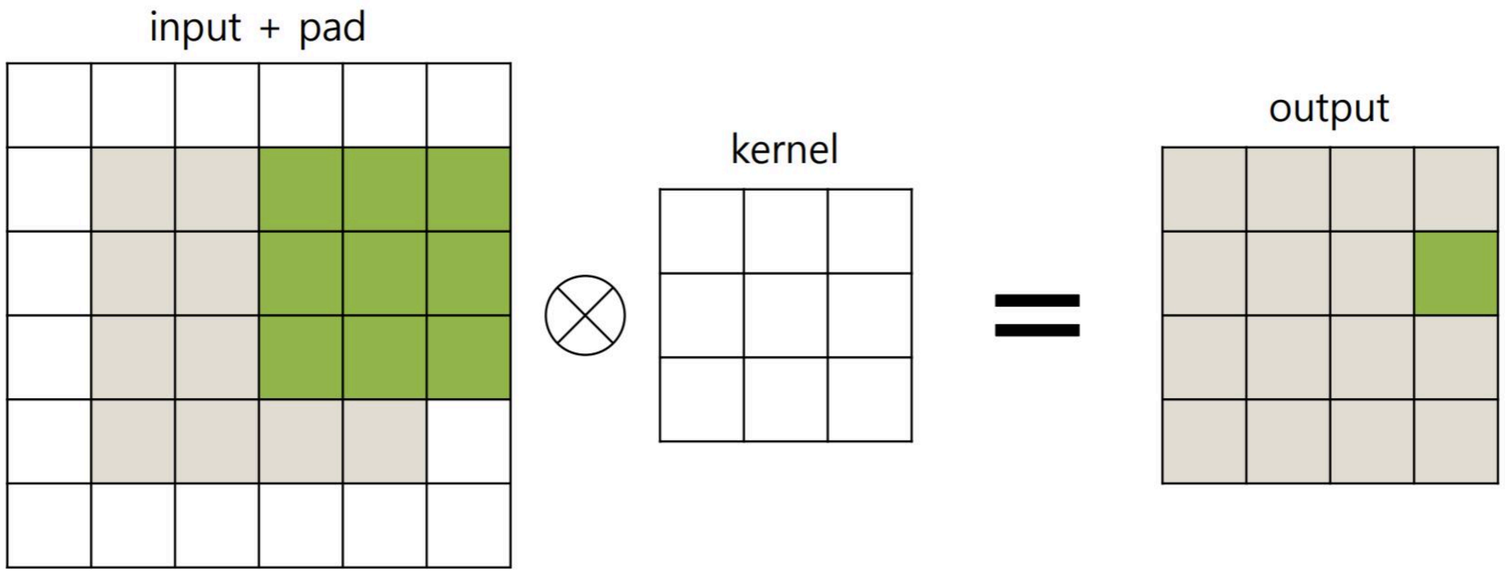
Convolutional Neural Network (with Pad)



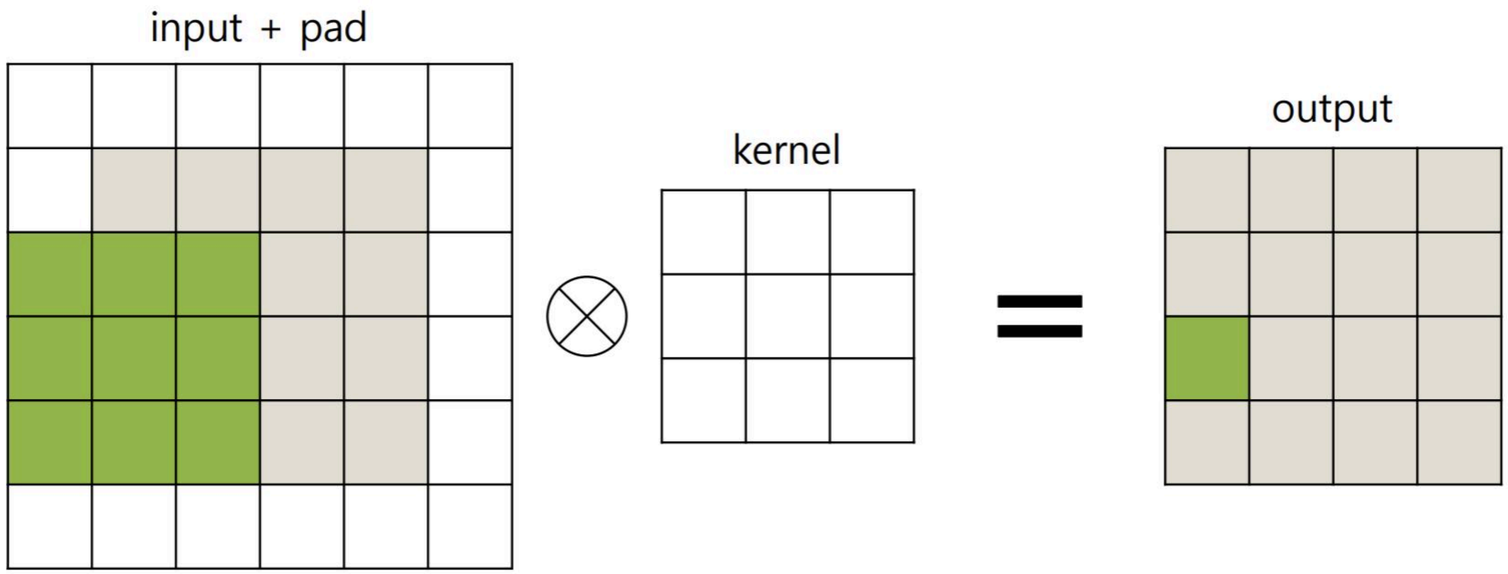
Convolutional Neural Network (with Pad)



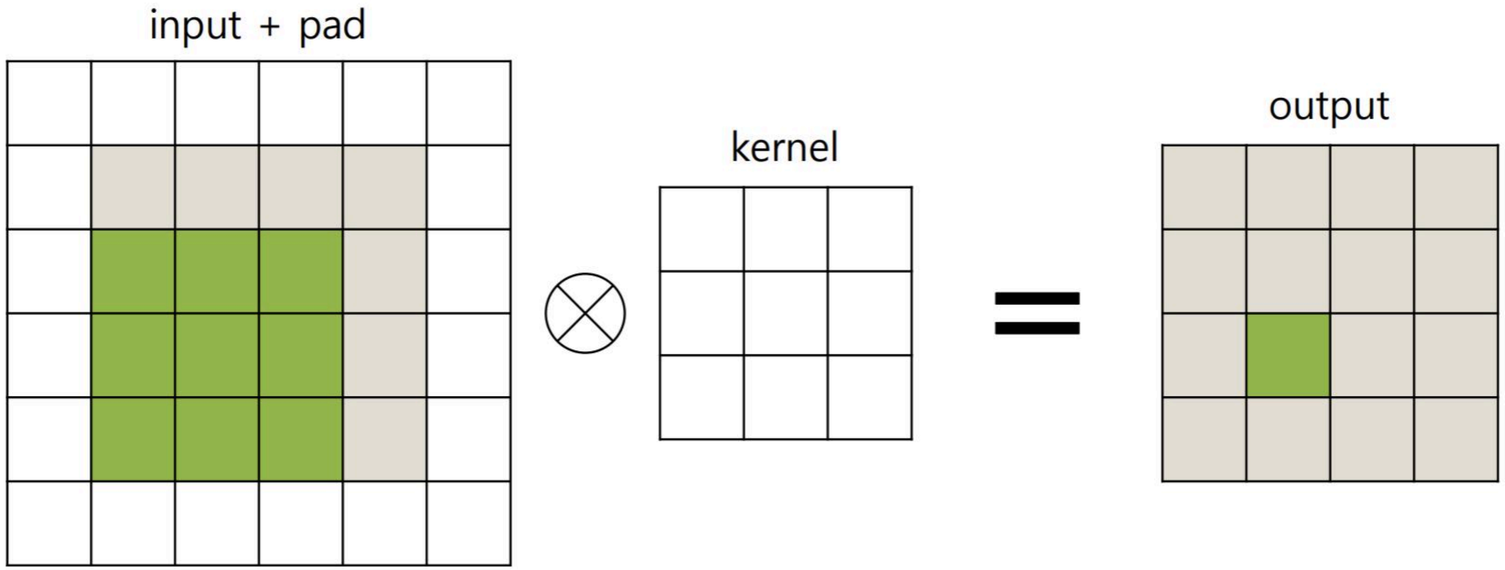
Convolutional Neural Network (with Pad)



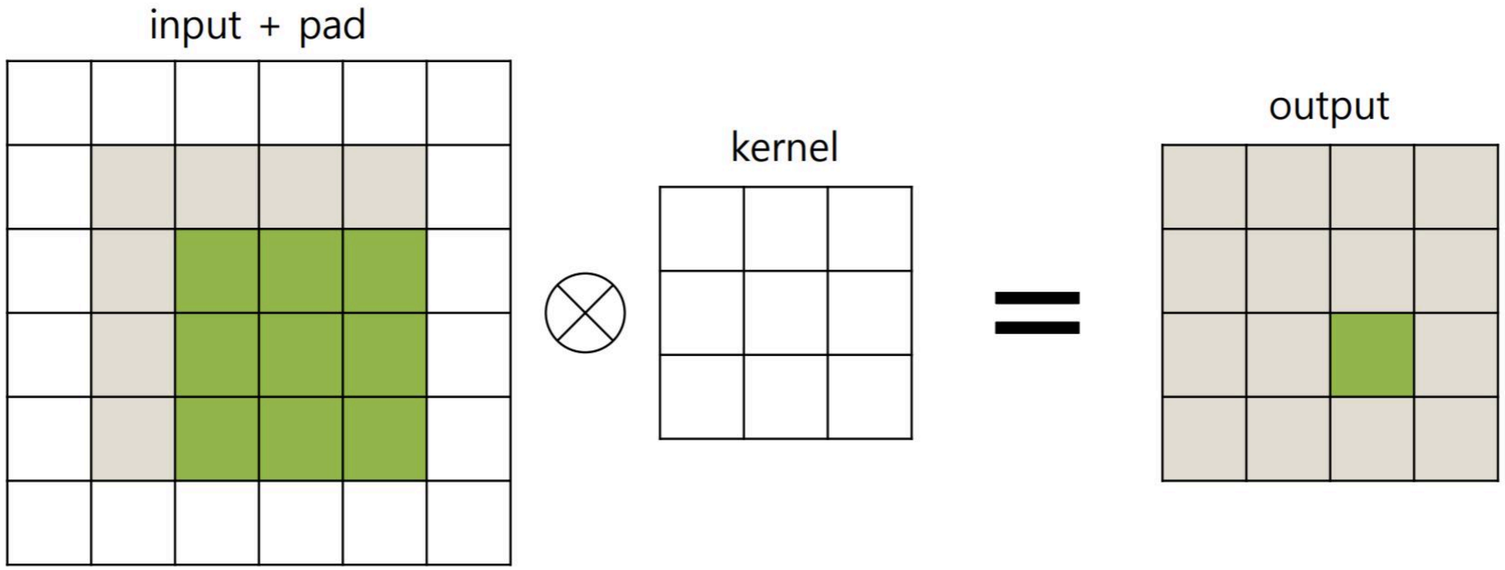
Convolutional Neural Network (with Pad)



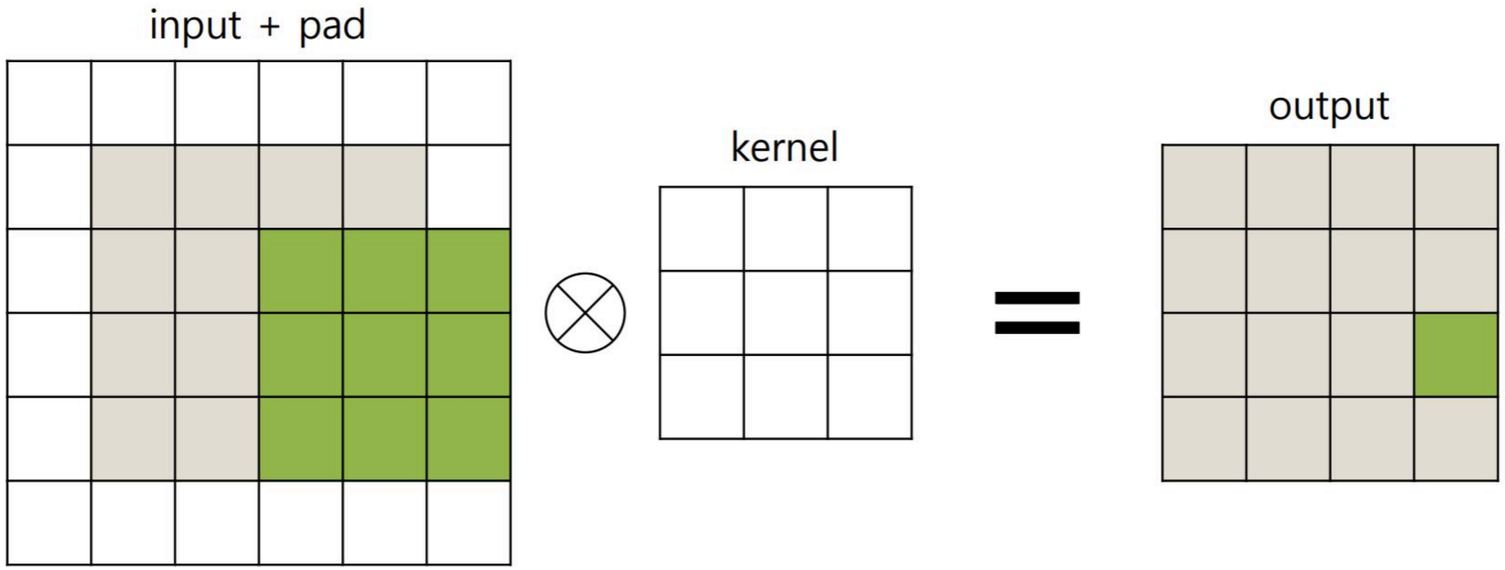
Convolutional Neural Network (with Pad)



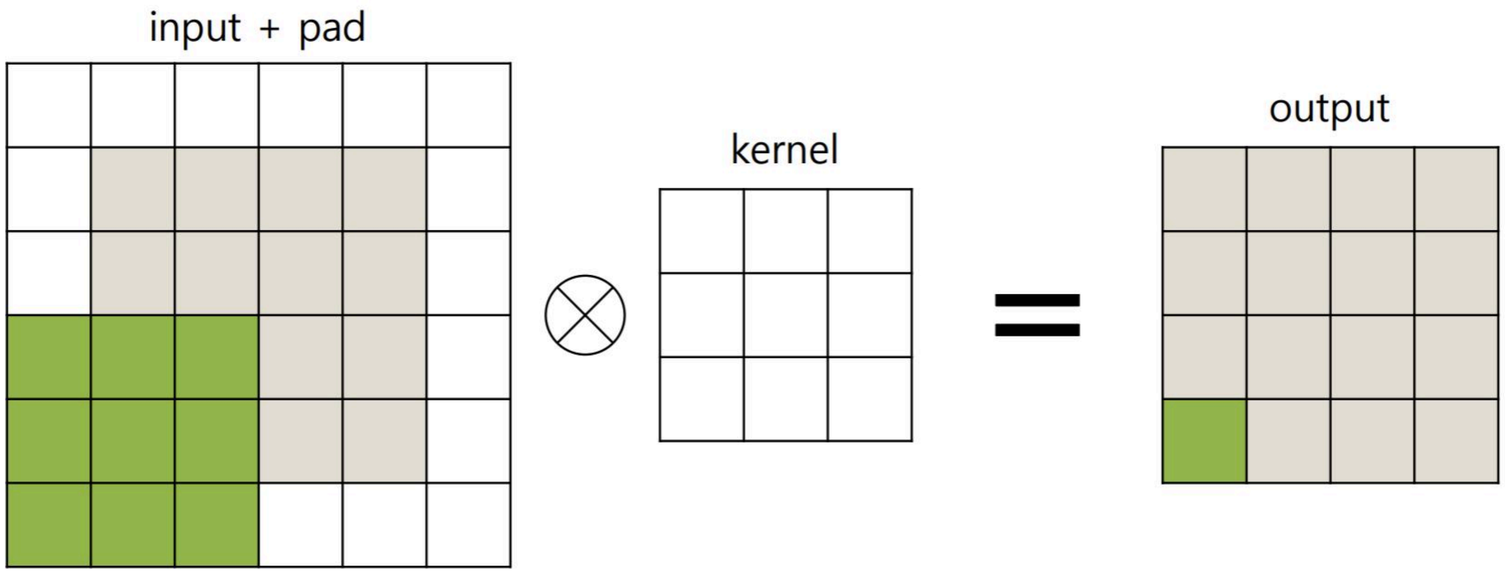
Convolutional Neural Network (with Pad)



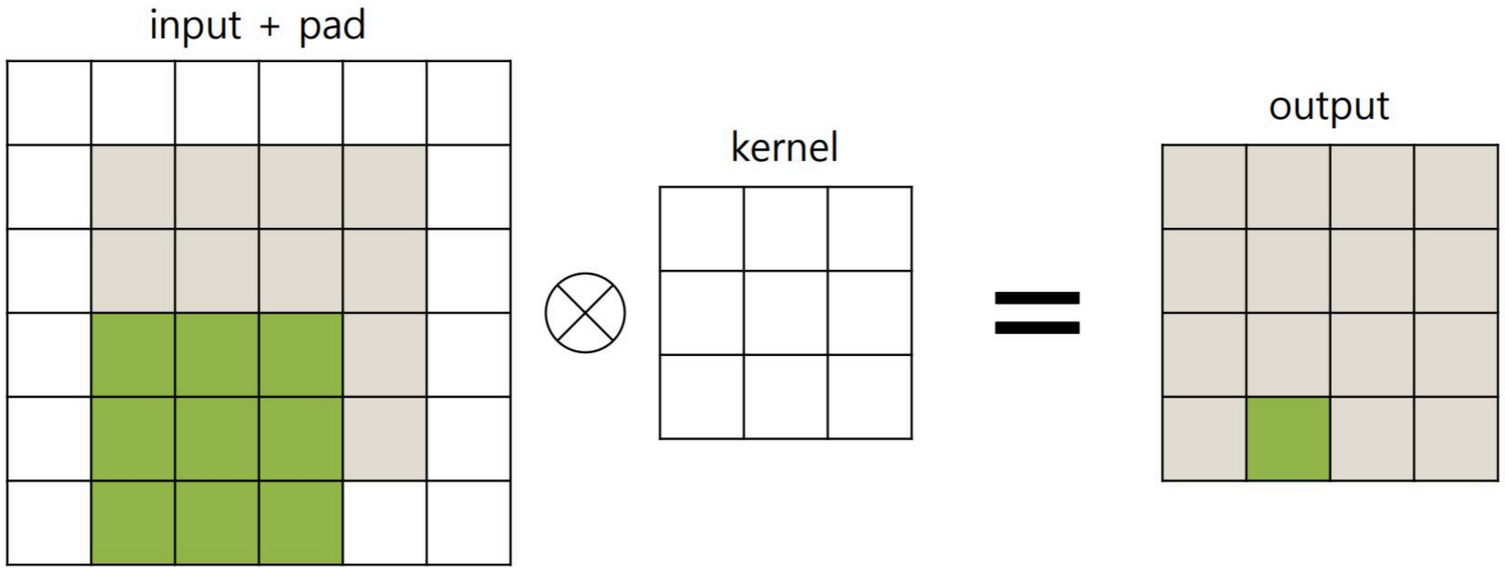
Convolutional Neural Network (with Pad)



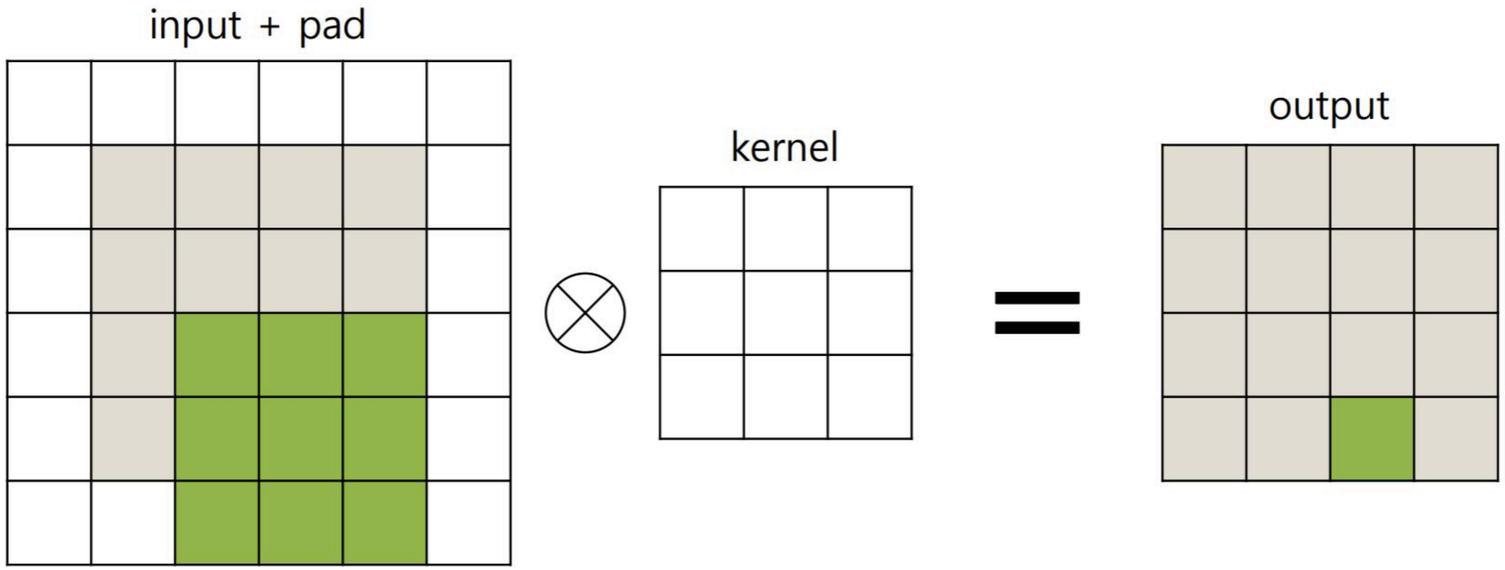
Convolutional Neural Network (with Pad)



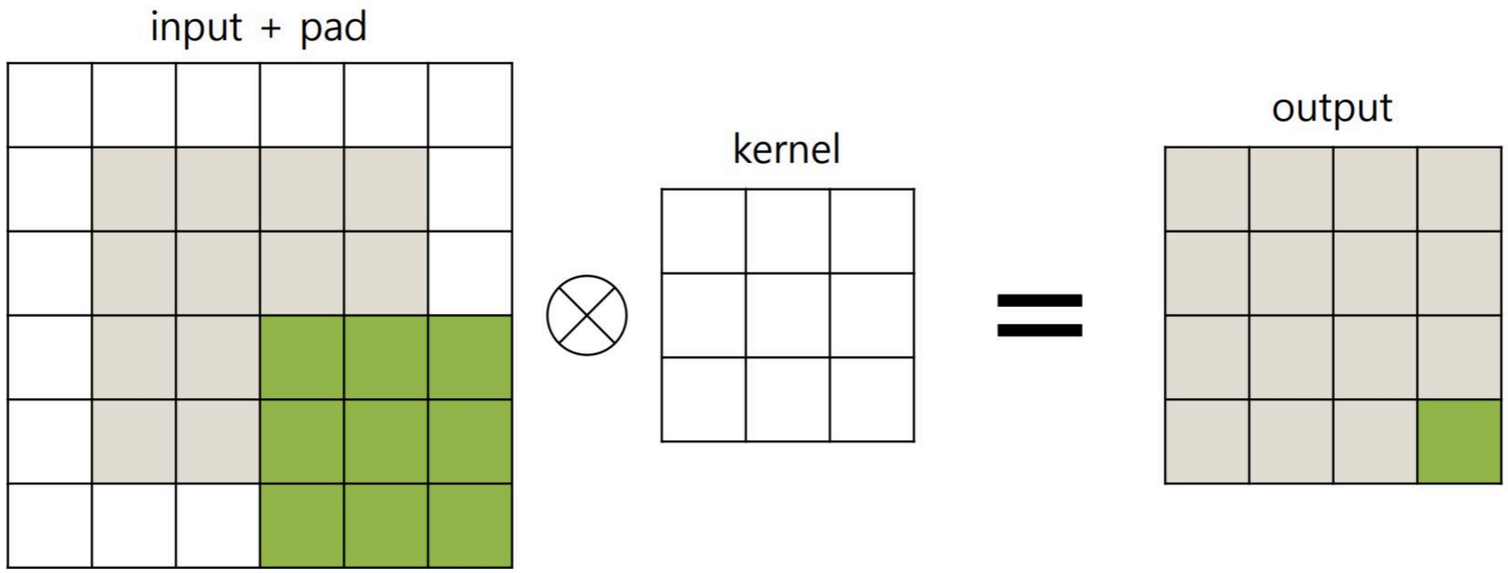
Convolutional Neural Network (with Pad)



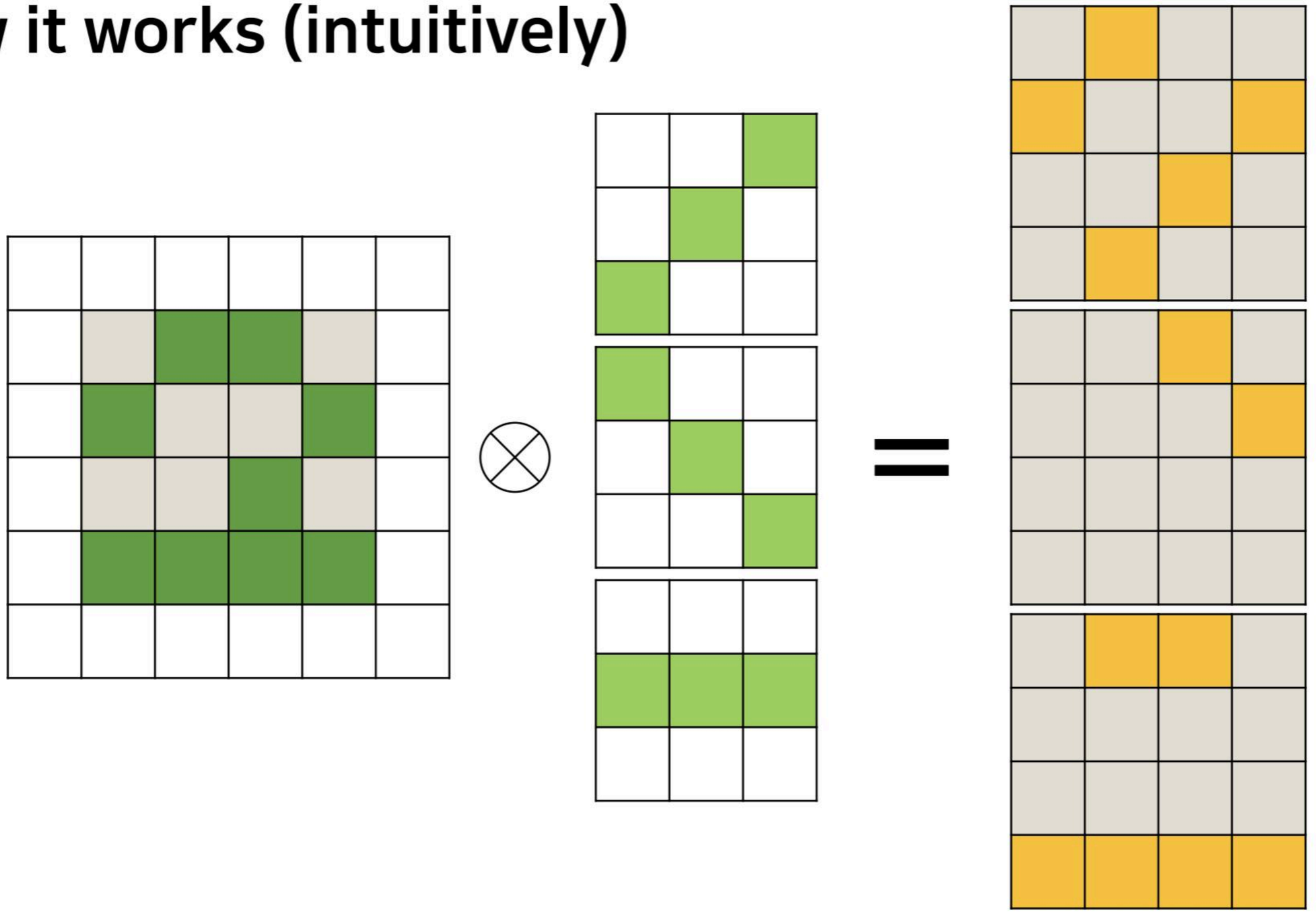
Convolutional Neural Network (with Pad)



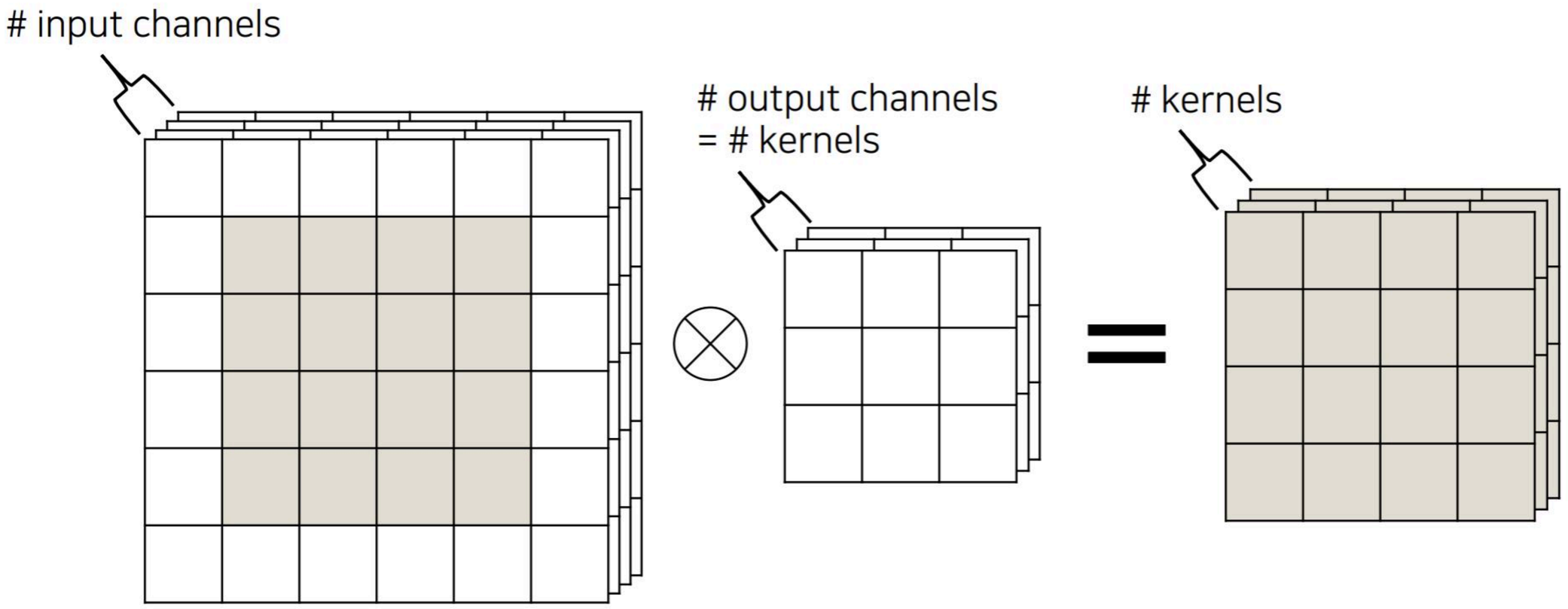
Convolutional Neural Network (with Pad)



How it works (intuitively)



Input & Output Channels

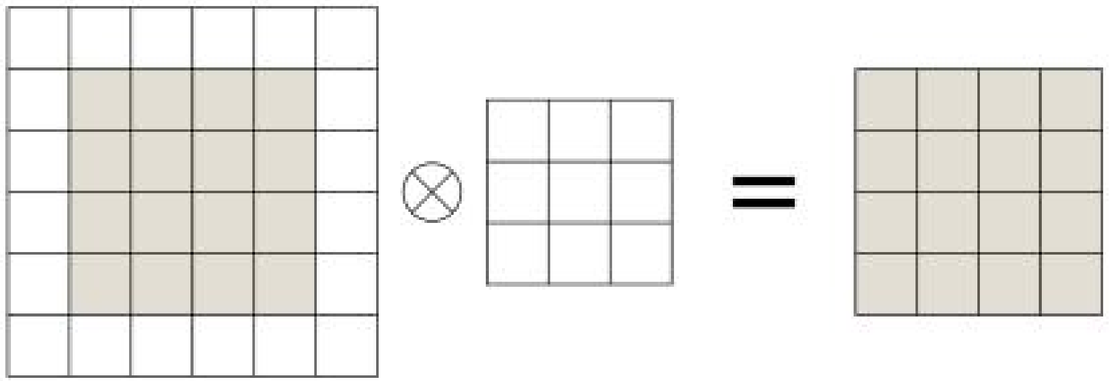


e.g. input channels of color image: Red, Green, Blue

Dimension Reduction

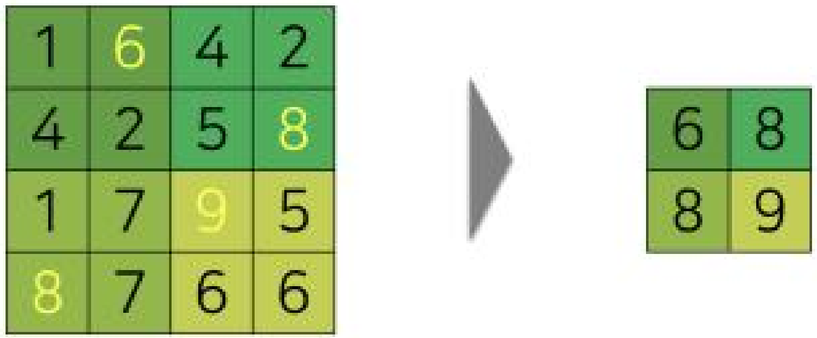
- 고차원의 sparse한 데이터를 저차원 공간에 mapping
  - 그 과정에서 복잡하게 얽혀있는 데이터를 풀어냄

- 하지만 3X3 + 1 padding 과 같은 convolusion layer는 입출력 텐서의 크기가 같음



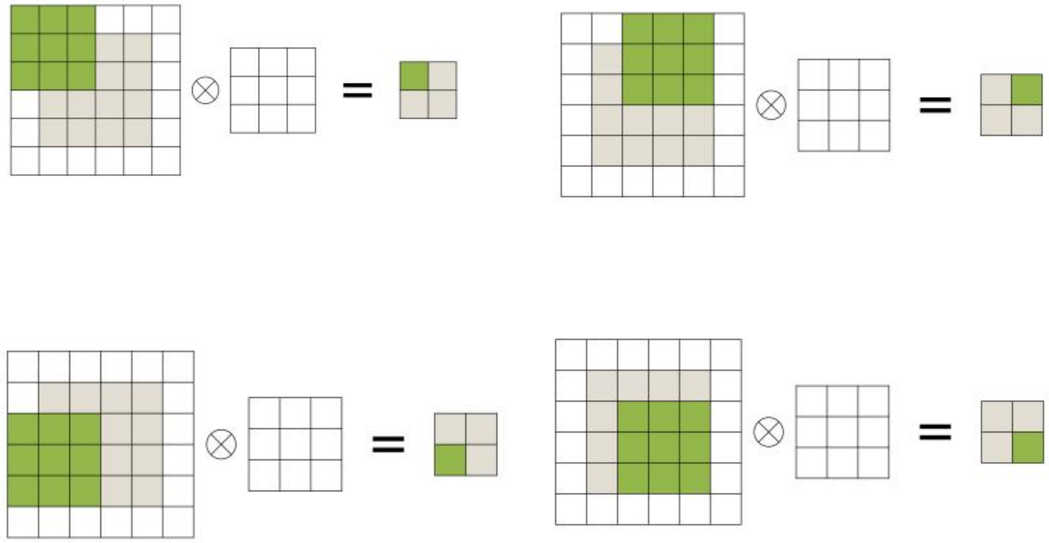
Max-pooling

- Down Sampling 기법
- 별도의 max-pooling layer를 활용



Stride

- 같은 convolution layer에서 동작



CNN의 특징

- FC Layer에 비해 입출력 크기가 계산이 까다로워, 네트워크 구성이 어렵다

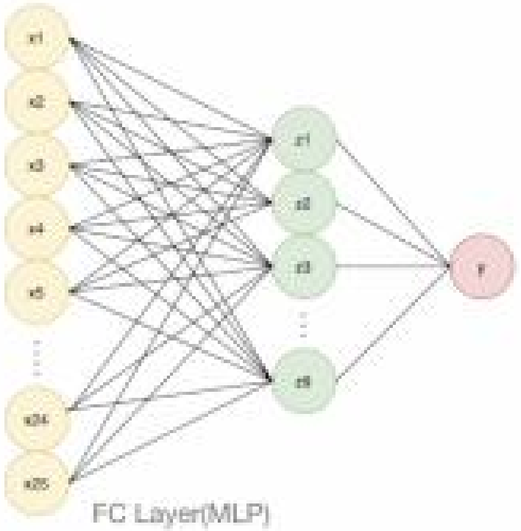
$$y = \mathrm{conv2d}(x),$$

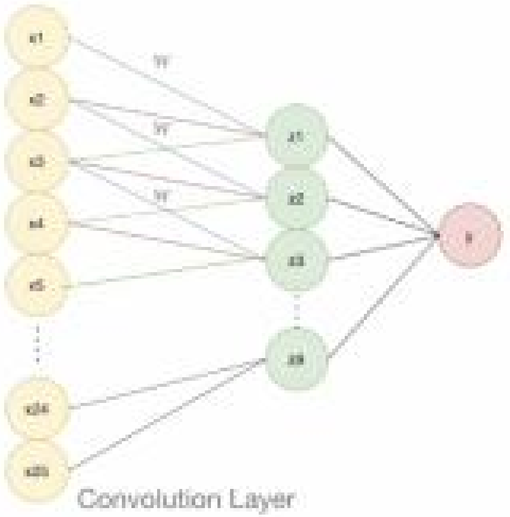
$$\text{where} \begin{cases} |x| = (b, c_{\text{in}}, x_{\text{height}}, x_{\text{width}}) \\ |y| = (y_{\text{height}}, y_{\text{width}}) \\ \quad = \left(b, c_{\text{out}}, \left\lfloor \frac{x_{\text{height}} + 2 \times p_{\text{height}} - (k_{\text{height}} - 1) - 1}{s_{\text{height}}} + 1 \right\rfloor, \left\lfloor \frac{x_{\text{width}} + 2 \times p_{\text{width}} - (k_{\text{width}} - 1) - 1}{s_{\text{width}}} + 1 \right\rfloor \right). \end{cases}$$

  - Feature의 위치에 구애 받지 않는다.
    - 패턴 자체를 학습
  - FC Layer에 비해 더 적은 weight를 가진다.
    - kernel을 재활용.
  - 병렬 계산 구성이 쉽다. GPU에서의 연산이 매우 빠르다
  - Convolution Layer는 자동으로 탐지해야 할 패턴을 추출
    - 내부 kernel은 해당 패턴을 추출하기 위한 형태로 자동구성

CNN : Parameter sharing/sparsity

- Convolution Layer는 Fully Connected보다 sparse한 연결
- 재사용되는 Parmeter를 업데이트. Kernel이 적용될 때마다 gradient가 더해짐.
- 적은 수의 파라미터를 재사용하는 특성으로 인해, 학습 속도와 효율성이 매우 뛰어난 구조.





CNN Architecture

- **CNN Block**

1. **3×3 Convolution Layer(pad)**
2. **ReLU**
3. **Batch Normalization**
4. **3×3 Convolution Layer(pad) (+ Stride size(2 × 2)**
5. **ReLu**
6. **Batch Normalization**
7. **(+Max pooling layer)**