

CV DEEP LEARNING WITH PYTORCH

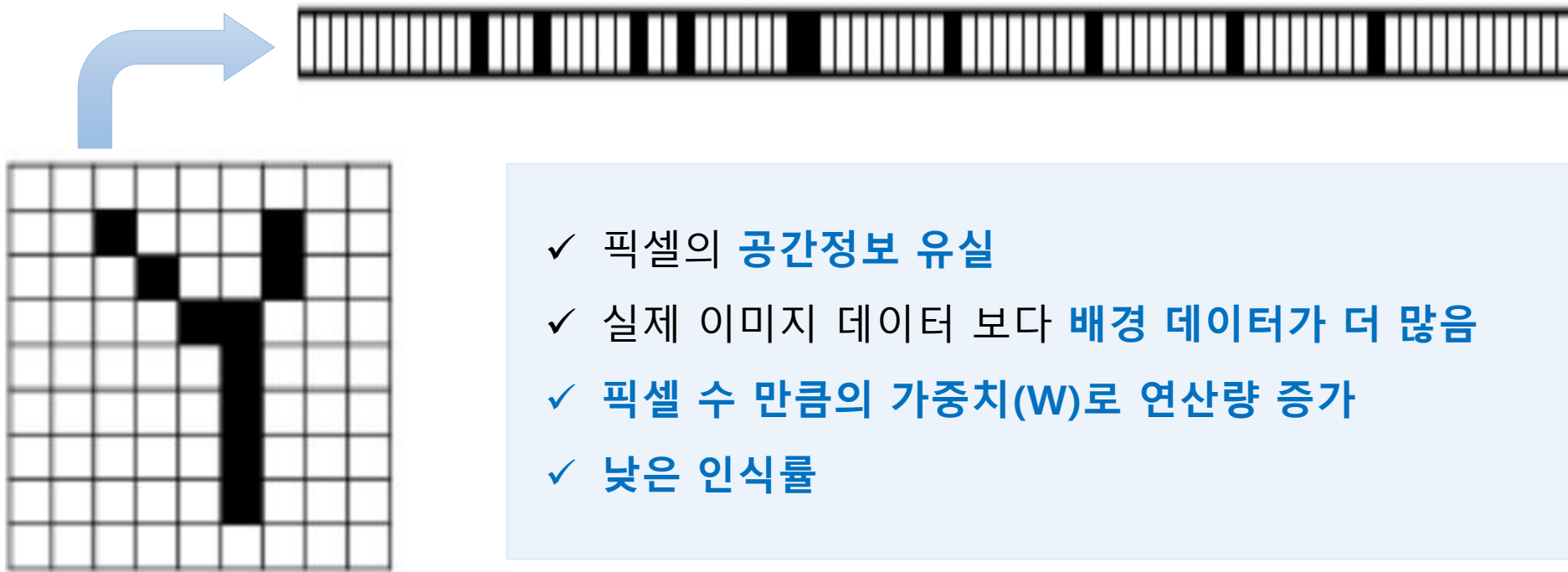
PART III

CONVOLUTIONAL NEURAL NETWORKS

CONVOLUTIONAL NEURAL NETWORKS

3

◆ DNN (MLP) 이미지 인식

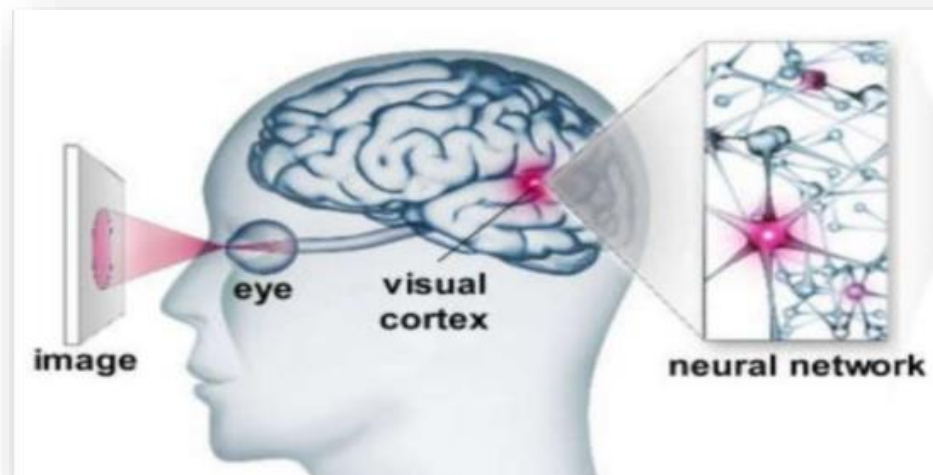


CONVOLUTIONAL NEURAL NETWORKS

4

◆ 사람/동물 시각 인식

- 동물의 시각피질(Visual Cortex) 구조에서 영감
- 시각 자극이 **1차 시각피질을 통해서 처리 → 2차 시각피질 경유 → 3차 시각피질**
- 여러 영역 통과하며 모여진 정보를 계층적으로 처리
- 추상적인 특징 추출하여 시각 인식

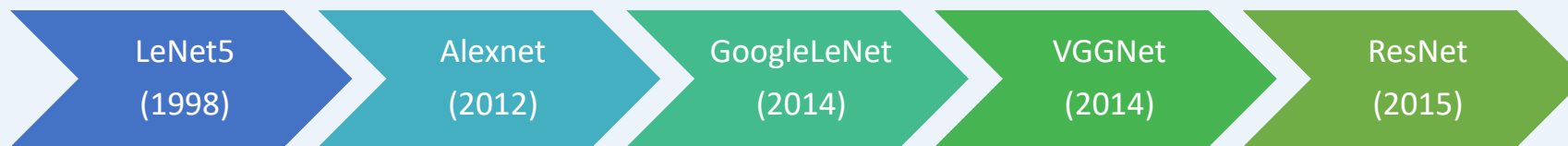


CONVOLUTIONAL NEURAL NETWORKS

5

◆ CNN 합성곱신경망

- DNN의 한 종류로 컴퓨터 **비전**, **시각적 이미지 인식** 주로 사용
- **텍스트 처리** 등 **여러 다른 분야에도 다양하게 활용**
- **LeNet-5**은 **1998년** Yann LeCun 교수가 발표한 CNN 알고리즘으로 지속적인 연구와 발전 진행, 특히 2010년 초중반에 많은 발전



CONVOLUTIONAL NEURAL NETWORKS

6

◆ CNN 합성곱신경망

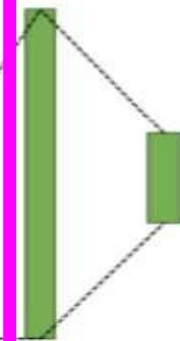
❖ 입력 데이터의 형상을 유지하며 특징 추출 후 분류

- 전반부 : 3차원 이미지 입력 받아 특징 추출
- 후반부 : 특징 입력 받아 분류

전반부



후반부



CONVOLUTIONAL NEURAL NETWORKS

7

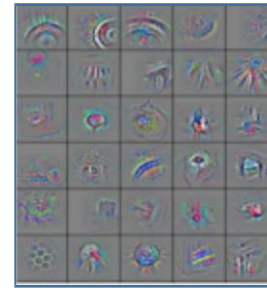
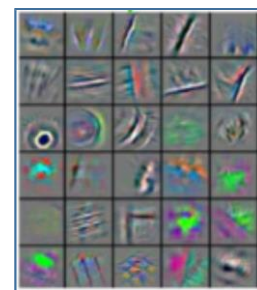
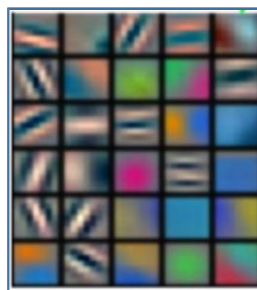
◆ Convolution Layer 합성곱층

❖ 동작

- 이미지 위를 일정 간격으로 이동하며 **특징(정보)**를 하나씩 추출
- 위에서 아래로 전체 이동으로 특징(정보)를 모은 **특징맵 출력 기능 Layer**



입력



특징맵

◆ Convolution Layer 합성곱층

■ 커널/필터/마스크

- 중치로 구성되며 일반적으로 3x3, 5x5 크기
- 너무 큰 커널은 특징 추출 부족
- 이미지 위를 일정 간격 이동

■ 스트라이드(stride)

- 커널의 이동 방향 및 크기
- 기본 : 왼쪽 -> 오른쪽 1칸, 위 -> 아래 1칸

CONVOLUTIONAL NEURAL NETWORKS

9

◆ Convolution Layer 합성곱층

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

4		



1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0

4	3	



1	1	1 _{x1}	0 _{x0}	0 _{x1}
0	1	1 _{x0}	1 _{x1}	0 _{x0}
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1	1	0
0	1	1	0	0

4	3	4

■ 스트라이드(stride)

- 이동 방향 : 왼쪽상단 → 오른쪽하단
- 기본 : 왼쪽 -> 오른쪽 1칸

CONVOLUTIONAL NEURAL NETWORKS

10

◆ Convolution Layer 합성곱층

1	1	1	0	0
0 _{x1}	1 _{x0}	1 _{x1}	1	0
0 _{x0}	0 _{x1}	1 _{x0}	1	1
0 _{x1}	0 _{x0}	1 _{x1}	1	0
0	1	1	0	0

4	3	4
2		



1	1	1	0	0
0	1 _{x1}	1 _{x0}	1 _{x1}	0
0	0 _{x0}	1 _{x1}	1 _{x0}	1
0	0 _{x1}	1 _{x0}	1 _{x1}	0
0	1	1	0	0

4	3	4
2	4	



1	1	1	0	0
0	1	1 _{x1}	1 _{x0}	0 _{x1}
0	0	1 _{x0}	1 _{x1}	1 _{x0}
0	0	1 _{x1}	1 _{x0}	0 _{x1}
0	1	1	0	0

4	3	4
2	4	3

■ 스트라이드(stride)

- 이동 방향 : 왼쪽상단 → 오른쪽하단
- 기본 : 왼쪽 -> 오른쪽 1칸

CONVOLUTIONAL NEURAL NETWORKS

11

◆ Convolution Layer 합성곱층

1	1	1	0	0
0	1	1	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0 _{x0}	0 _{x1}	1 _{x0}	1	0
0 _{x1}	1 _{x0}	1 _{x1}	0	0

4	3	4
2	4	3
2		



1	1	1	0	0
0	1	1	1	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0 _{x0}	1 _{x1}	1 _{x0}	0
0	1 _{x1}	1 _{x0}	0 _{x1}	0

4	3	4
2	4	3
2	3	



1	1	1	0	0
0	1	1	1	0
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1 _{x0}	1 _{x1}	0 _{x0}
0	1	1 _{x1}	0 _{x0}	0 _{x1}

4	3	4
2	4	3
2	3	4

■ 스트라이드(stride)

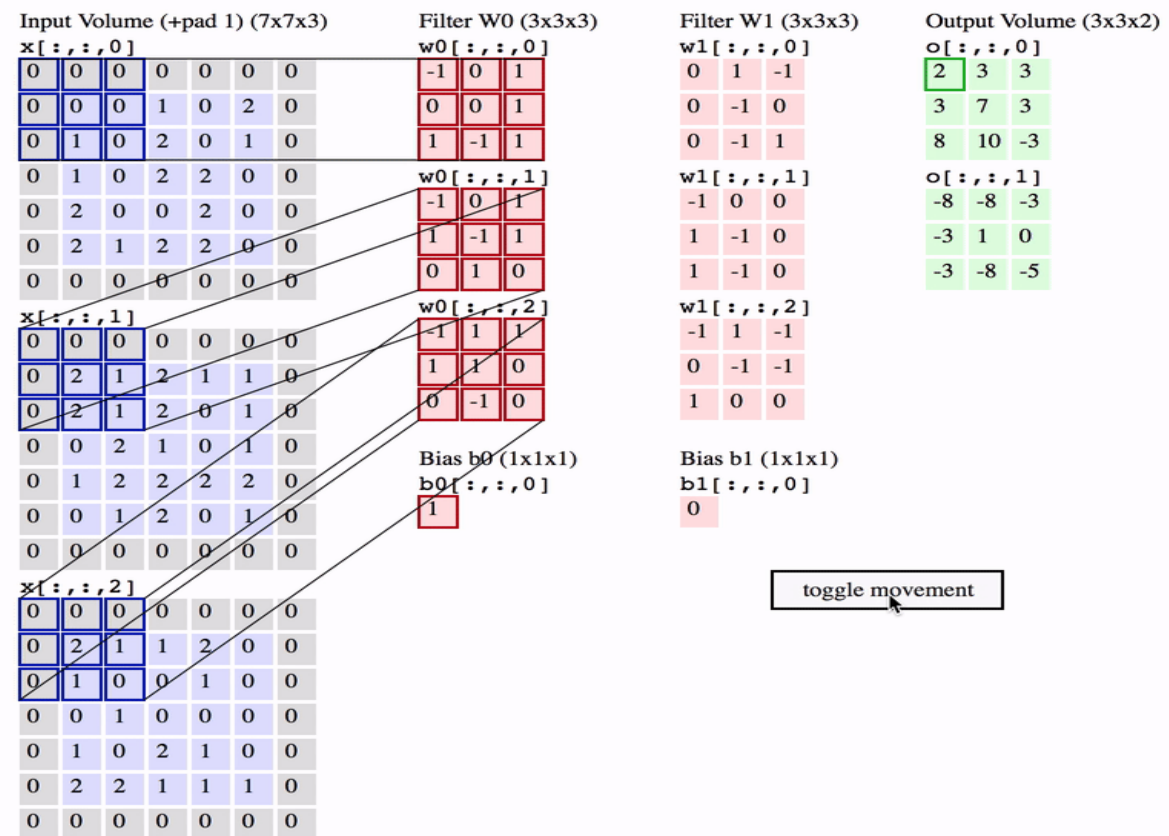
- 이동 방향 : 왼쪽상단 → 오른쪽하단
- 기본 : 왼쪽 -> 오른쪽 1칸

CONVOLUTIONAL NEURAL NETWORKS

12

◆ Convolution Layer 합성곱층

❖ Feature Map 추출 동작



CONVOLUTIONAL NEURAL NETWORKS

13

◆ Convolution Layer 합성곱층

- 1채널 그레이스케일 이미지

$$(1 \times 1) + (0 \times 0) + (0 \times 1) + (0 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times 1) + (0 \times 0) + (1 \times 1) = 3$$



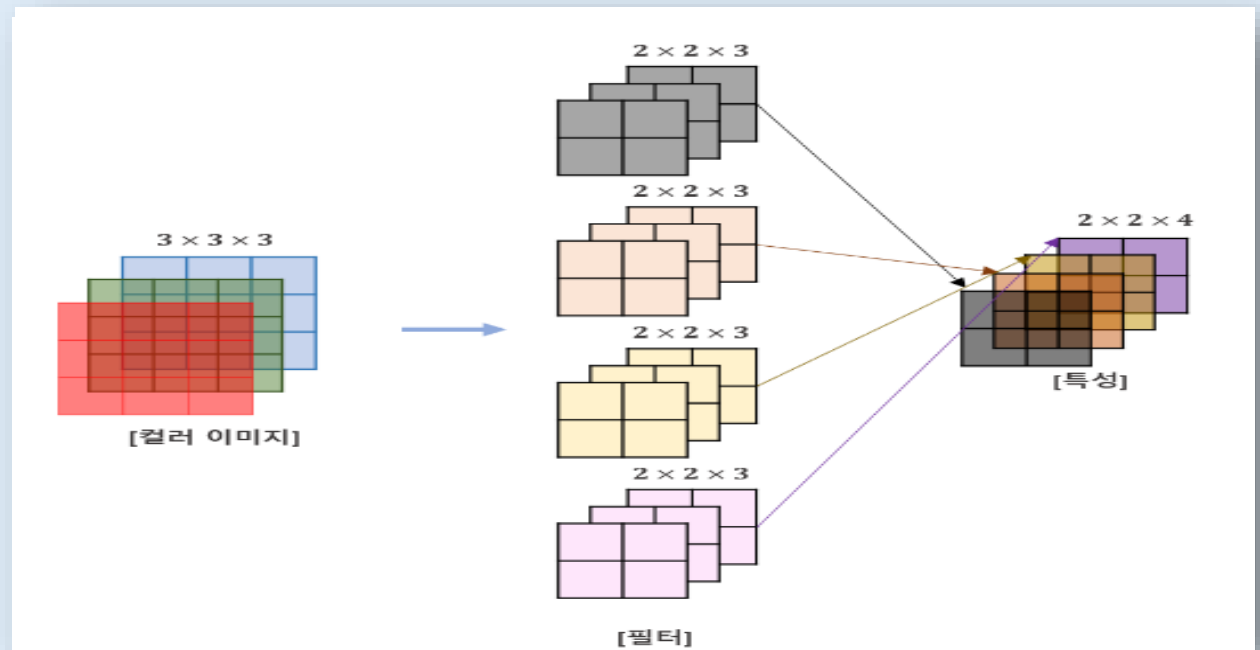
CONVOLUTIONAL NEURAL NETWORKS

14

◆ Convolution Layer 합성곱층

■ 3채널 컬러 이미지

		165	187	209	58	7
	14	125	233	201	98	159
253	144	120	251	41	147	204
67	100	32	241	23	165	30
209	118	124	27	59	201	79
210	236	105	169	19	219	156
35	178	199	197	4	14	218
115	104	34	111	19	196	
32	69	231	203	74		



◆ Convolution Layer 합성곱층

▪ 패딩(Padding)

- 커널(필터) 이동 시 좌우상하 모서리 부분 특징 추출 안됨
- 입력 데이터 사면을 특정값(0)으로 채운 후 합성곱층 진행
- 종류 : Valid Padding, Same Padding

CONVOLUTIONAL NEURAL NETWORKS

16

◆ Convolution Layer 합성곱층

- 패딩(Padding) - Valid
 - 입력과 출력 특징맵 Shape 다름

1	1	1	0	0
0	1	1	1	0
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1 _{x0}	1 _{x1}	0 _{x0}
0	1	1 _{x1}	0 _{x0}	0 _{x1}

4	3	4
2	4	3
2	3	4

CONVOLUTIONAL NEURAL NETWORKS

17

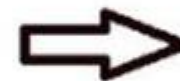
◆ Convolution Layer 합성곱층

- 패딩(Padding) - Same

- 입력과 출력 특징맵 Shape 같음
- 입력 데이터 사면을 특정값(0)으로 채운 후 진행

x1	x0	x1
x0	x1	x0
x1	x0	x1

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



2	2	3	1	1
1	4	3	4	1
2	2	5	3	3
1	2	3	4	1
1	2	3	1	1

◆ Convolution Layer 합성곱층

▪ Conv1D

- 커널/필터가 **시간을 축으로 좌우로만 이동**할 수 있는 합성곱
- 데이터 ▶ 시계열데이터
- 입력 ▶ 2D/3D : [batch,] embedding, sequence_length
- 출력 ▶ 2D/3D : [batch,] embedding, sequence_length
- 사례 ▶ 문장 분류, 소리, 신호 평활화

CONVOLUTIONAL NEURAL NETWORKS

19

◆ Convolution Layer 합성곱층

▪ Conv1D

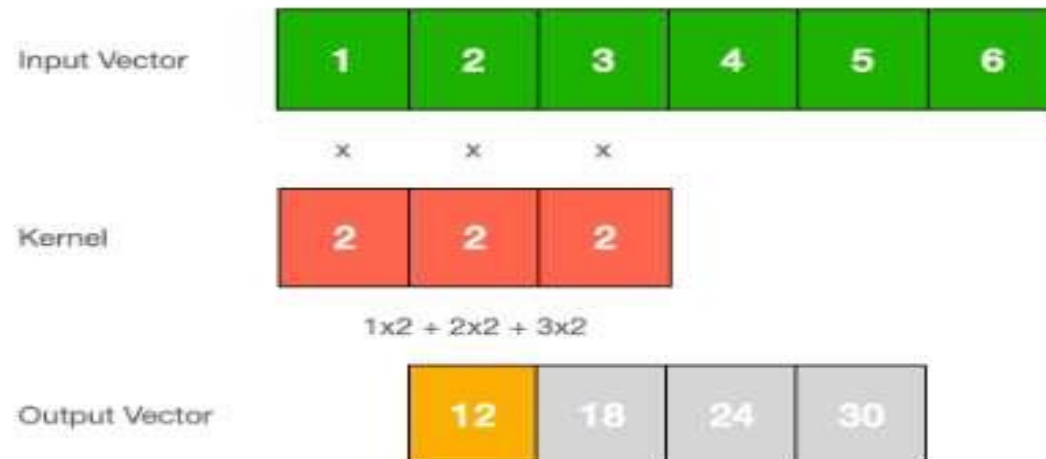


CONVOLUTIONAL NEURAL NETWORKS

20

◆ Convolution Layer 합성곱층

▪ Conv1D

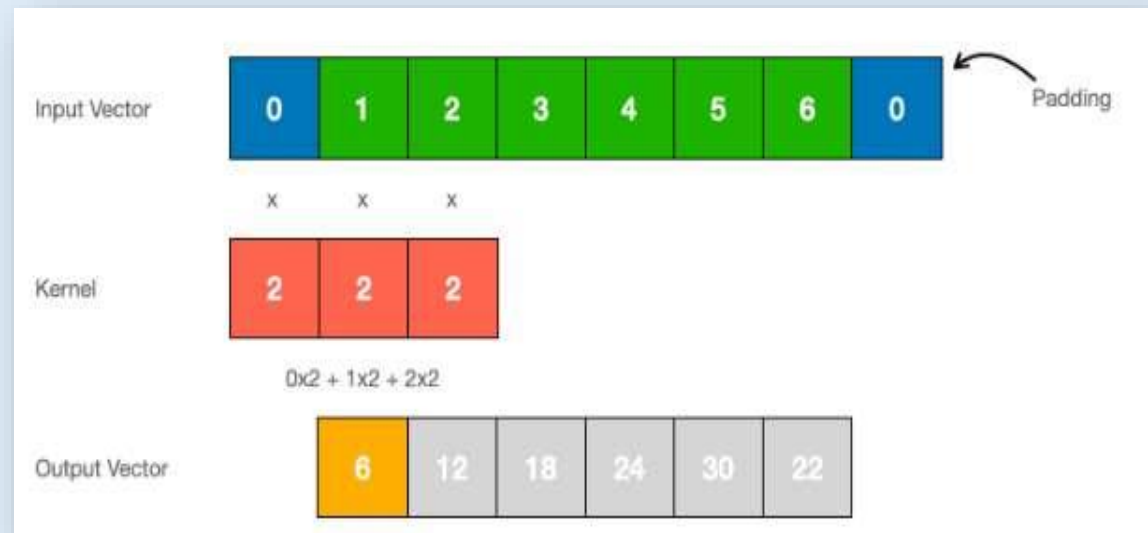


CONVOLUTIONAL NEURAL NETWORKS

21

◆ Convolution Layer 합성곱층

▪ Conv1D



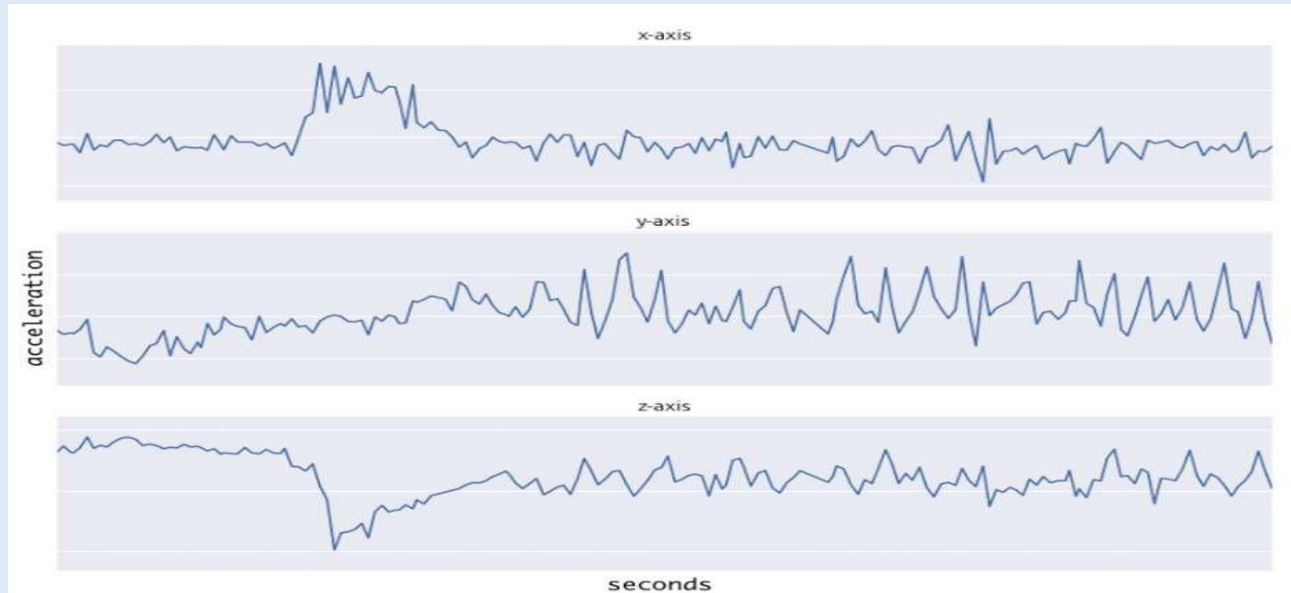
CONVOLUTIONAL NEURAL NETWORKS

22

◆ Convolution Layer 합성곱층

- Conv1D

[예] 가속도계 센싱 데이터



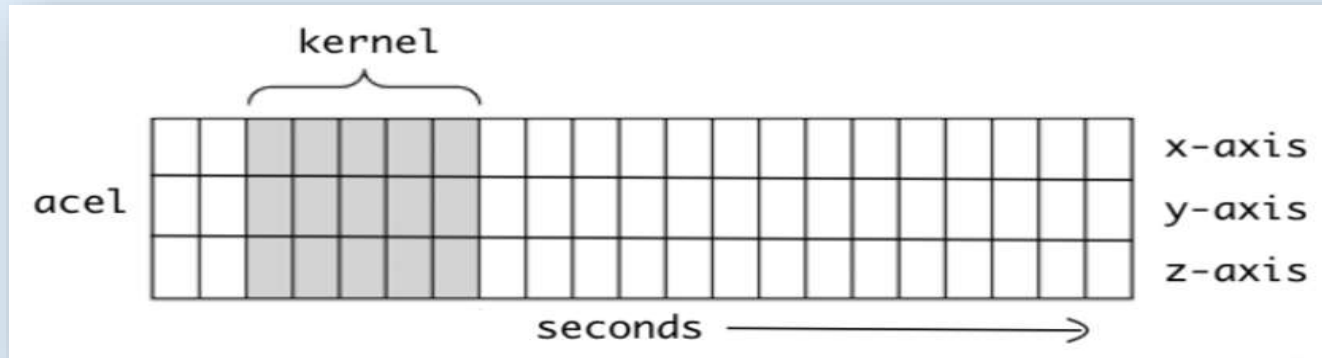
CONVOLUTIONAL NEURAL NETWORKS

23

◆ Convolution Layer 합성곱층

▪ Conv1D

[예] 가속도계 센싱 데이터



◆ Convolution Layer 합성곱층

▪ Conv2D

- 커널/필터가 **시간을 축으로 좌우로만 이동**할 수 있는 합성곱
- 데이터 ▶ 이미지
- 입력 ▶ 3D/4D : [batch,] in_channels, in_height, in_width,
- 출력 ▶ 3D/4D : [batch,] out_channels, filter_height, filter_width
- 사례 ▶ 객체 인식

◆ Convolution Layer 합성곱층

▪ Conv3D

- 커널/필터가 **세 개 방향으로 이동**할 수 있는 합성곱
- 데이터 ▶ 3D 이미지(MRI, CT 등등) , 비디오
- 입 력 ▶ 4D/5D : [batch,] in_channels, in_depth, in_height, in_width
- 출 력 ▶ 4D/5D : [batch,] out_channels, filter_depth, filter_height, filter_width
- 사 례 ▶ CCTV 이상행동 감지

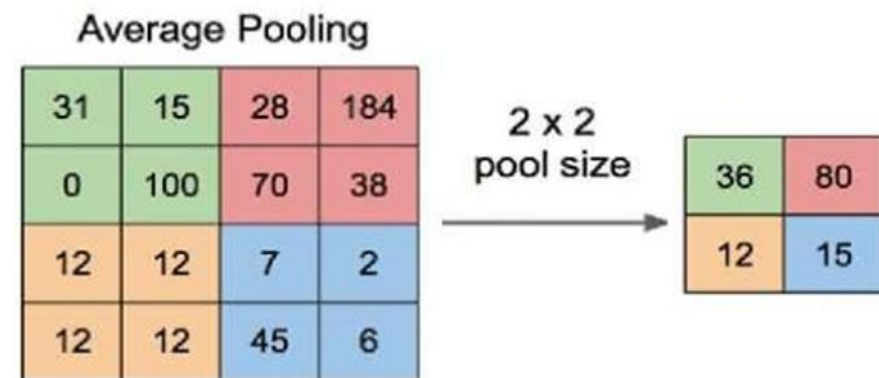
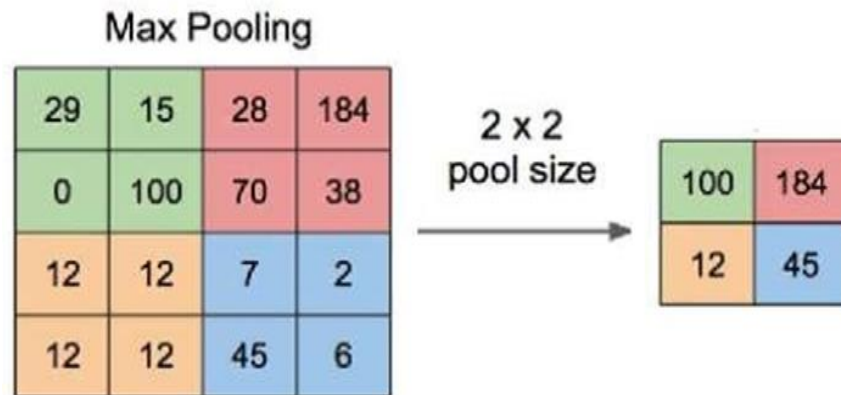
◆ Polling Layer 풀링층

- 합성곱 층(합성곱 연산 + 활성화 함수) 다음에 풀링 층 추가
- 특성 맵을 다운샘플링하여 특성 맵의 크기 줄이는 풀링 연산 진행
- 합성곱층과 달리 커널이 중첩되지 않음
- 커널 크기 : 2x2, 2의 배수
- 종류 : 최대 풀링(max pooling), 평균 풀링(average pooling)

CONVOLUTIONAL NEURAL NETWORKS 27

◆ Polling Layer 폴링층

- 통상적으로 (2,2) 크기 / 스트라이드 역시 (2,2)
- 겹치는 부분 없이 연산

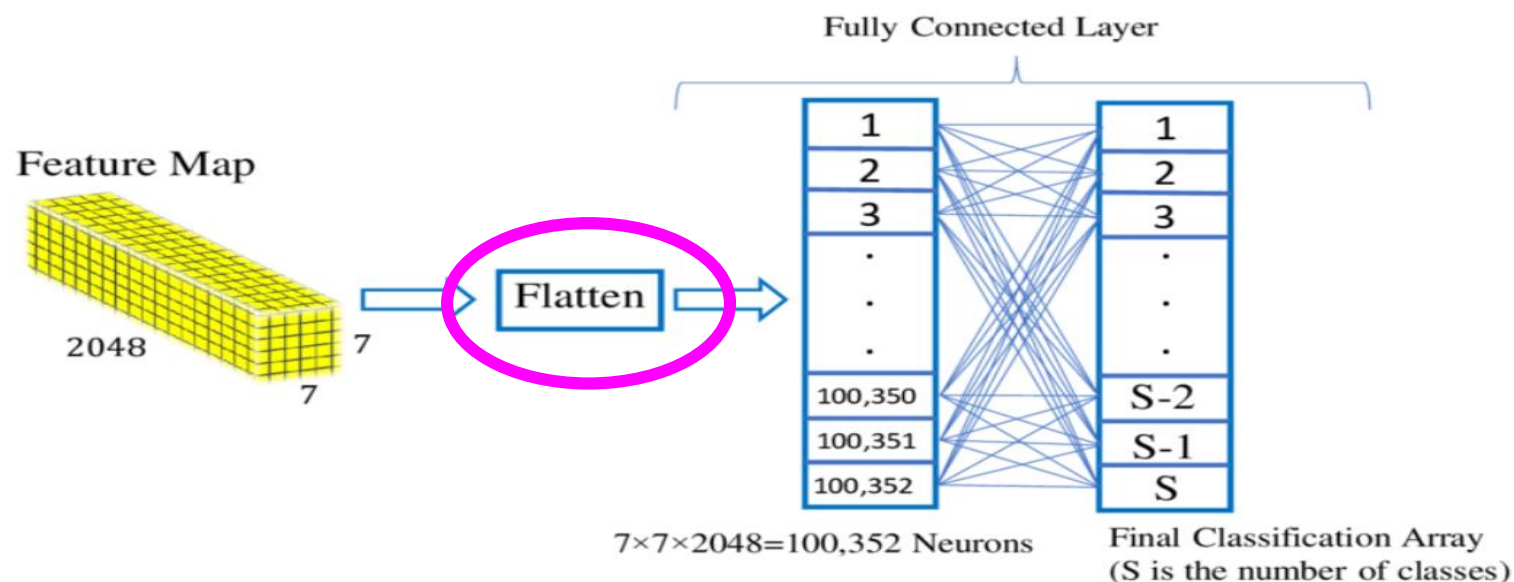


CONVOLUTIONAL NEURAL NETWORKS

28

◆ Flatten Layer 벡터화층

- 마지막 output layer을 1차원 벡터 데이터 변형하는 층
- 여러 layer 통해 입력 이미지에서 얻어온 특이점 데이터 1차원 데이터로 변형

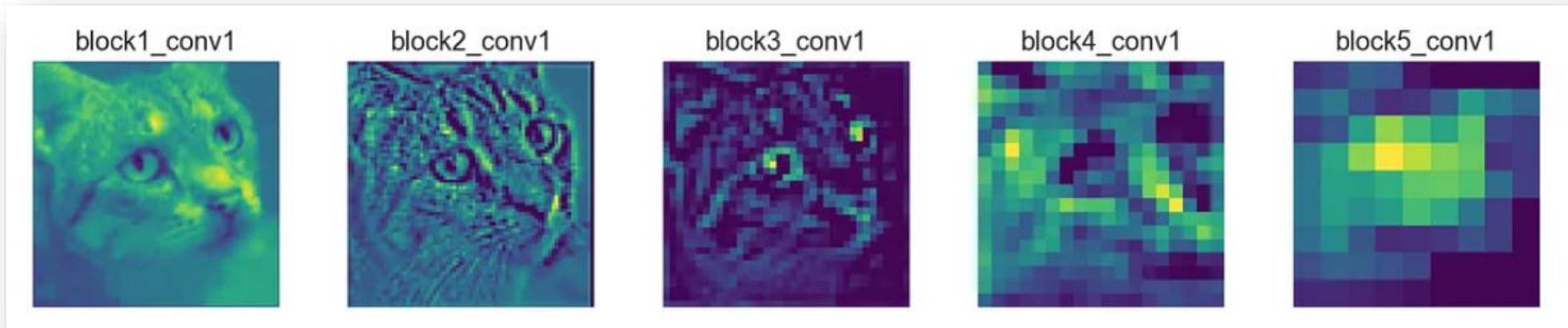


CONVOLUTIONAL NEURAL NETWORKS

29

◆ CNN 합성곱신경망

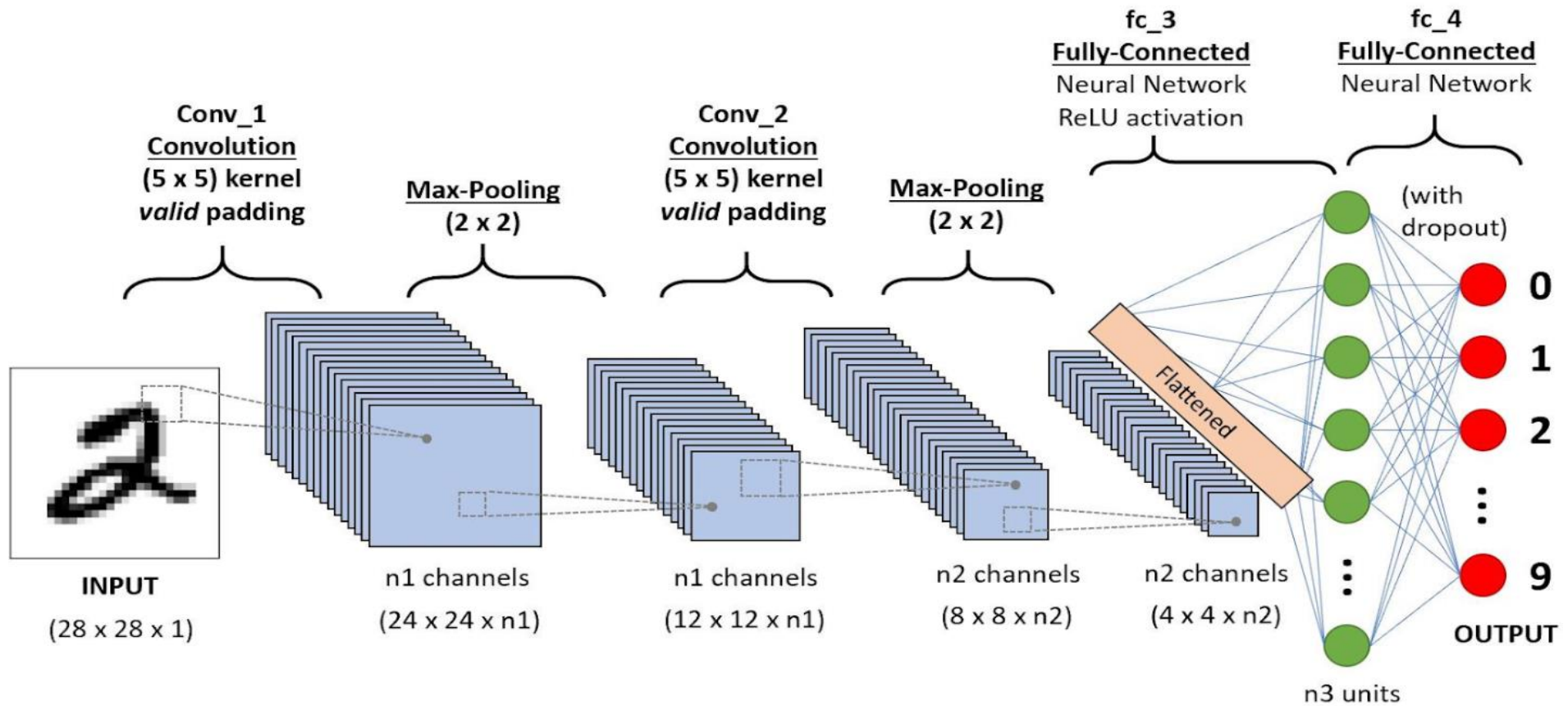
▪ Feature Map 시각화



CONVOLUTIONAL NEURAL NETWORKS

30

◆ CNN 합성곱신경망



◆ Torch.nn.Convolution Layers

- 구현된 Class

`nn.Conv1d`

Applies a 1D convolution over an input signal composed of several input planes.

`nn.Conv2d`

Applies a 2D convolution over an input signal composed of several input planes.

`nn.Conv3d`

Applies a 3D convolution over an input signal composed of several input planes.

`nn.ConvTranspose1d`

Applies a 1D transposed convolution operator over an input image composed of several input planes.

`nn.ConvTranspose2d`

Applies a 2D transposed convolution operator over an input image composed of several input planes.

`nn.ConvTranspose3d`

Applies a 3D transposed convolution operator over an input image composed of several input planes.

CONVOLUTIONAL NEURAL NETWORKS

32

◆ Torch.nn.Conv1D

```
nn.Conv1D( in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,  
           groups=1, bias=True, padding_mode='zeros', device=None, dtype=None )
```

N : batch size **C** : Embedding dimension **L** : sequence length

INPUT SHAPE	(N, C _{in} , L _{in}) 또는 (C _{in} , L _{in})	OUTPUT SHAPE	(N, C _{out} , L _{out}) 또는 (C _{out} , L _{out})
-------------	---	--------------	---

in_channels : 임베딩 차원 수

out_channels : 커널 차원 수

kernel_size : 커널 사이즈, int 또는 tuple.

stride : stride 사이즈, 기본값 1, int 또는 tuple.

padding : padding 사이즈, 기본값 0, 'valid', 'same'
int 또는 tuple.

padding_mode : 기본 값 'zeros'

dilation : 데이터와 커널 사이 간격 사이즈 조절

groups : 입력 채널과 출력 채널 사이 관계

group=1 : 모든 입력은 모든 출력과 convolution 연산 [기본]

group=2 : 입력을 2 그룹으로 나누어 각각 convolution 연산, 결과 concatenation

group=in_channels : 각각의 인풋 채널이 각각 아웃풋 채널 대응

CONVOLUTIONAL NEURAL NETWORKS

33

◆ Torch.nn.Conv2D

```
nn.Conv2D( in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,  
           groups=1, bias=True, padding_mode='zeros', device=None, dtype=None )
```

N : batch size

C : image channel

H : height, rows

W : width, columns

INPUT

(N, C_{in}, H_{in}, W_{in}) 또는 (C_{in}, H_{in}, W_{in})

OUTPUT

(N, C_{out}, H_{out}, W_{out}) 또는 (C_{out}, H_{out}, W_{out})

in_channels : 입력 채널 수, 흑백 이미지 1, 컬러 이미지 3

out_channels : 출력 커널 수

kernel_size : 커널 사이즈, int 또는 tuple.

stride : stride 사이즈, 기본값 1

padding : padding 사이즈, 기본값 0, 'valid', 'same'

padding_mode : 기본 값 'zeros'

dilation : 커널 사이 간격 사이즈 조절

groups : 입력 층의 그룹 수 설정, 입력 채널 수를 그룹 수로 분리, 입출력 그룹 연산

bias : bias 값을 설정 할 지, 말지를 결정, 기본 값은 True

CONVOLUTIONAL NEURAL NETWORKS

34

◆ Torch.nn.Conv3D

```
nn.Conv3D( in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,  
           groups=1, bias=True, padding_mode='zeros', device=None, dtype=None )
```

N : batch size

C : image channel

D : Depth

H : height, rows

W : width, columns

INPUT

(N, C_{in}, D_{in}, H_{in}, W_{in}) 또는 (C_{in}, D_{in}, H_{in}, W_{in})

OUTPUT

(N, C_{out}, D_{out}, H_{out}, W_{out}) 또는 (C_{out}, D_{out}, H_{out}, W_{out})

in_channels : 입력 채널 수, 흑백 이미지 1, 컬러 이미지 3

out_channels : 출력 커널 수

kernel_size : 커널 사이즈, int 또는 tuple.

stride : stride 사이즈, 기본값 1

padding : padding 사이즈, 기본값 0, 'valid', 'same'

padding_mode : 기본 값 'zeros'

dilation : 커널 사이 간격 사이즈 조절

groups : 입력 층의 그룹 수 설정, 입력 채널 수를 그룹 수로 분리, 입출력 그룹 연산

bias : bias 값을 설정 할 지, 말지를 결정, 기본 값은 True

PART III

TORCHVISION

◆ PyTorch 비전 라이브러리

- 파이토치에서 제공하는 **이미지 데이터셋들이 모여 있는 패키지**
- MNIST, ImageNet을 포함한 **유명한 데이터셋들을 제공**
- **모델 아키텍처 및 컴퓨터 비전 위한 이미지 변환 기능 제공**
 - 유명 이미지 데이터셋 로딩 기능 : CIFAR, COCO, MNIST, ImageNet
 - 미리 학습된(pre-trained) 이미지 분류 모델 제공 : VGG, ResNet, Inception
 - 이미지 전처리 기능 제공 : Transfrom 다양한 함수 제공
 - 다양한 유틸 함수 제공 : Utils

◆ 데이터 증강 Data Augmentation

- 데이터에 변형 가하여 **데이터 규모 증가 및 변형된 다양한 데이터 케이스 학습 제공**
- 효과 : **모델 과적합(overfitting) 방지**
 - Flip (Horizontal, Vertical)
 - Random Crop
 - Shear
 - Rotate
 - Zoom
 - Blur

◆ IMAGE DATASET

- 컴퓨터 비전 유명한 데이터셋
- <https://pytorch.org/vision/stable/datasets.html>

`Caltech101`(root[, target_type, transform, ...])

Caltech 101 Dataset.

`Caltech256`(root[, transform, ...])

Caltech 256 Dataset.

`CelebA`(root[, split, target_type, ...])

Large-scale CelebFaces Attributes (CelebA) Dataset Dataset.

`CIFAR10`(root[, train, transform, ...])

CIFAR10 Dataset.

◆ IMAGE DATASET

- torchvision.datasets.MNIST

```
import torch
from torchvision import datasets
from torchvision.transforms import ToTensor, Lambda

ds = datasets.MNIST( root="data",
                    train=True,
                    download=True,
                    transform=ToTensor(),
                    target_transform=Lambda(lambda y:
                                            torch.zeros(10, dtype=torch.float)
                                            .scatter_(0, torch.tensor(y), value=1))
                    )
```

◆ IMAGE DATASET

- 사용자 정의 데이터셋 관련 모듈

`DatasetFolder(root, loader[, extensions, ...])`

A generic data loader.

`ImageFolder(root, transform, ...)`

A generic data loader where the images are arranged in this way by default: .

`VisionDataset([root, transforms, transform, ...])`

Base Class For making datasets which are compatible with torchvision.

◆ IMAGE TRANSFORMS

- 이미지 **데이터의 전처리 및 데이터 증강** 위해 제공하는 모듈
- torchvision.transforms.XXX() 함수 기능

Resize	이미지 크기를 조절
RandomResizedCrop	무작위 자르고 크기 조절
RandomHorizontalFlip	무작위 수평으로 뒤집기
RandomVerticalFlip	무작위 수직으로 뒤집기
ToTensor	이미지 텐서로 변환
Normalize	이미지 정규화

RandomRotation	이미지 무작위 회전
RandomCrop	이미지 무작위 자름
Grayscale	이미지 흑백으로 변환
RandomSizedCrop	이미지 무작위 자르고 크기 조절
ColorJitter	이미지의 색상 무작위 조정

◆ IMAGE TRANSFORMS

- 이미지 데이터 전처리 기법들 구성 기능

```
from torchvision import transforms
```

```
transform = transforms.Compose(
```

```
[
```

```
    transforms.Resize(size=(512, 512)),
```

```
    transforms.ToTensor()
```

```
]
```

```
)
```

원하는 변형 조합 구성

PART III

TRANSFER LEARNING

◆ 전이 학습

❖ 사전 학습 (Pre-Training)

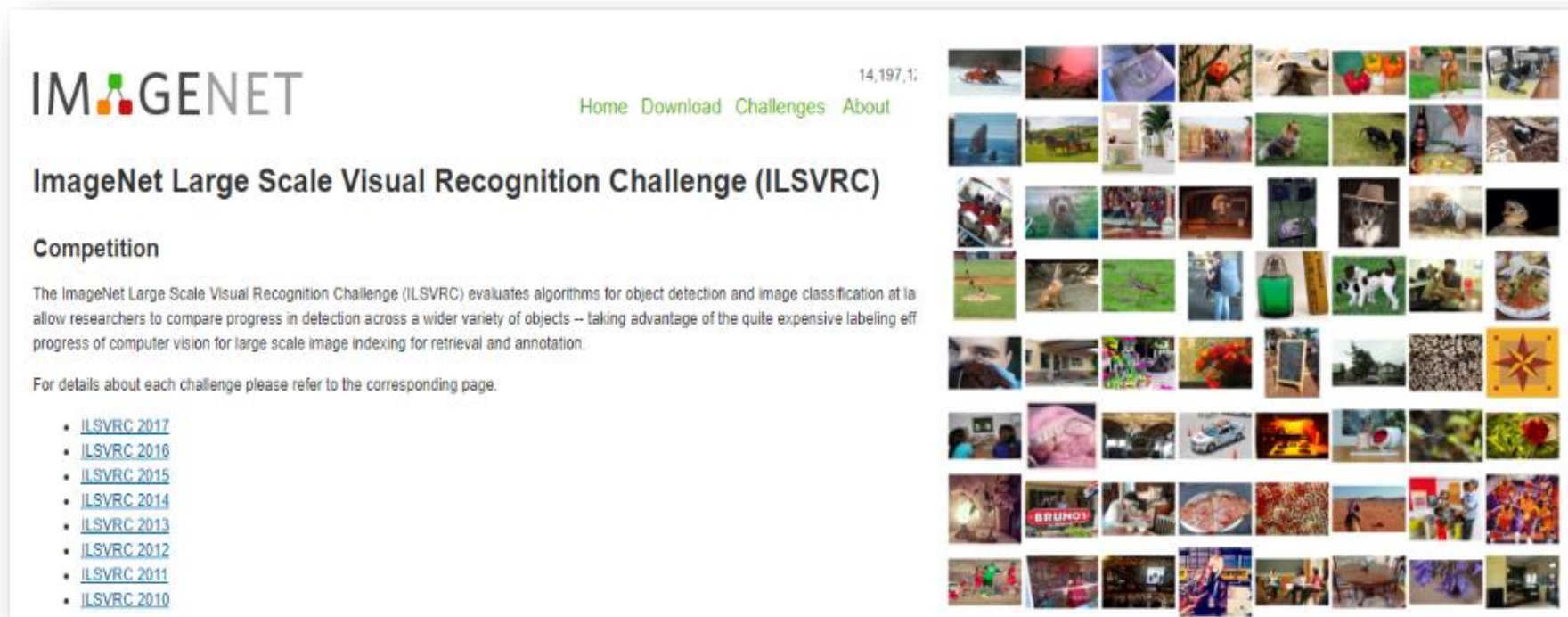
- 대량의 데이터로 학습이 되어 있는 모델의 가중치를 사용하는 방법
- 사전 학습 모델 가중치 사용 시 ➔ 학습 작업(task)는 하위 작업(downstream task)이 됨
- 사전 학습 데이터 셋 \supset 나의 학습 데이터 셋

❖ 파인 튜닝 (Fine-Tuning)

- 대량의 데이터로 학습이 되어 있는 모델의 가중치를 사용하는 방법
- 사전 학습 모델 가중치 사용 시 ➔ 학습 작업(task)는 하위 작업(downstream task)이 됨
- 사전 학습 데이터 셋 \supset 나의 학습 데이터 셋

◆ 전이 학습

❖ 이미지 분류 데이터셋 : <https://www.image-net.org/challenges/LSVRC/>



IMAGENET 14,197,11

Home Download Challenges About

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

Competition

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at a large scale to allow researchers to compare progress in detection across a wider variety of objects – taking advantage of the quite expensive labeling effort of computer vision for large scale image indexing for retrieval and annotation.

For details about each challenge please refer to the corresponding page.

- [ILSVRC 2017](#)
- [ILSVRC 2016](#)
- [ILSVRC 2015](#)
- [ILSVRC 2014](#)
- [ILSVRC 2013](#)
- [ILSVRC 2012](#)
- [ILSVRC 2011](#)
- [ILSVRC 2010](#)

The right side of the screenshot displays a grid of 100 small images, each representing a different class from the ImageNet dataset, such as animals, objects, and scenes.