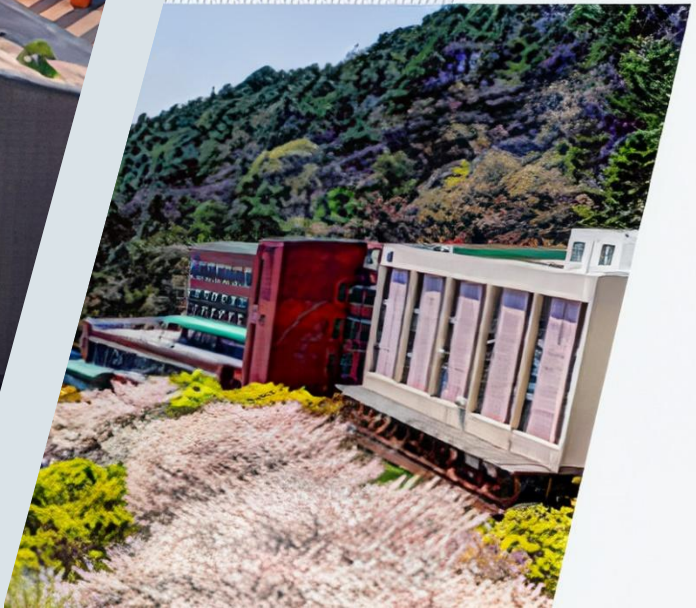
Dong-A Univ. (ISPL)

[실습] Convolutional Neural Network (CNN)

컴퓨터공학과

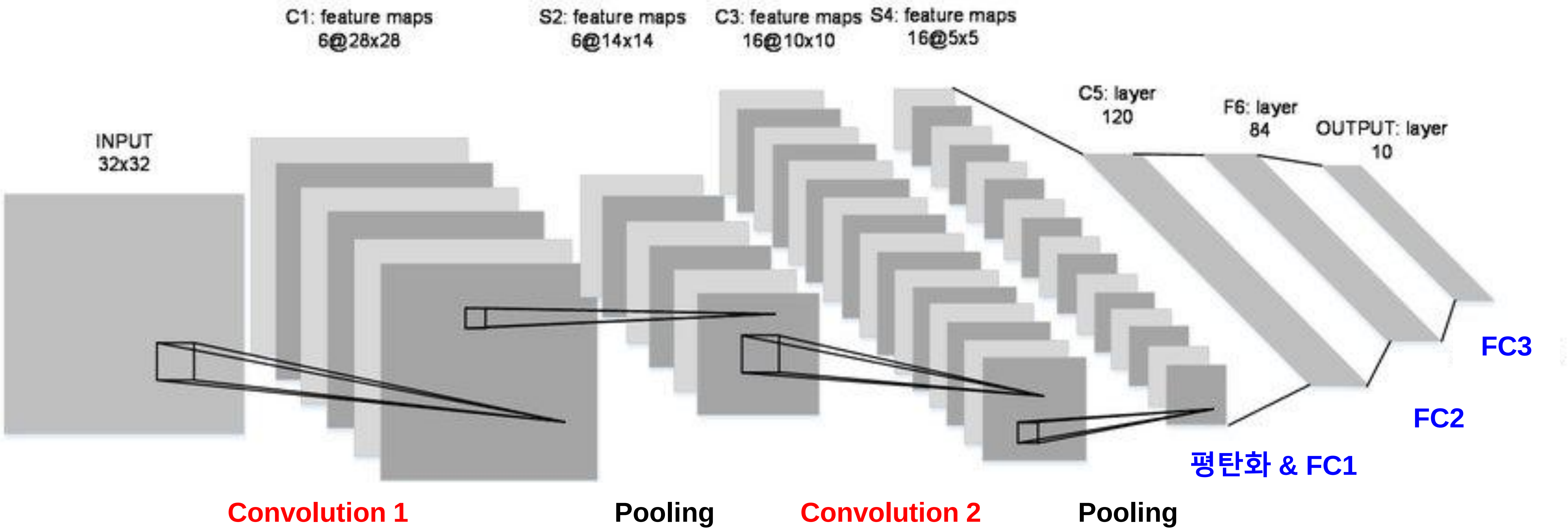
2024년 2학기 머신러닝



11주차 실습 (LeNet-5)

Convolutional Neural Network (CNN) 이론

- **CNN을 이용한 classification model 설계 시 주의사항**
  - 일반적으로 CNN의 feature map을 평탄화 한 이후 fully connected layer에 입력함



LeNet-5 구조

Ref.: [Proc. on IEEE 1998] Gradient-based learning applied to document recognition

LeNet-5 구조

C1	훈련해야할 파라미터 개수: (가중치*입력맵개수 + 바이어스)*특성맵개수 = (5*5*1 + 1)*6 = 156
S2	훈련해야할 파라미터 개수: (가중치 + 바이어스)*특성맵개수 = (1 + 1)*6 = 12
S4	훈련해야할 파라미터 개수: (가중치 + 바이어스)*특성맵개수 = (1 + 1)*16 = 32
C5	훈련해야할 파라미터 개수: (가중치*입력맵개수 + 바이어스)*특성맵 개수 = (5*5*16 + 1)*120 = 48120
F6	훈련해야할 파라미터 개수: 연결개수 = (입력개수 + 바이어스)*출력개수 = (120 + 1)*84 = 10164

Ref.: [Proc. on IEEE 1998] Gradient-based learning applied to document recognition

LeNet-5 C3

채널

입력

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

연속된 3장의 입력으로
convolution.
6개의 feature map 생성

연속된 4장의 입력으로
convolution.
6개의 feature map 생성

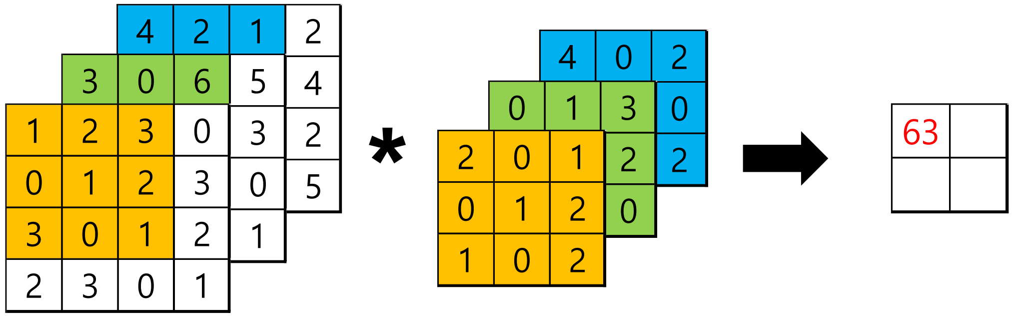
불연속된 4장의 입력으로
convolution.
3개의 feature map 생성

연속된 6장의 입력으로
convolution.
1개의 feature map 생성

Review – Convolutional layer

- **3D 이미지 (RGB) 입력에 대한 2D convolution 연산**

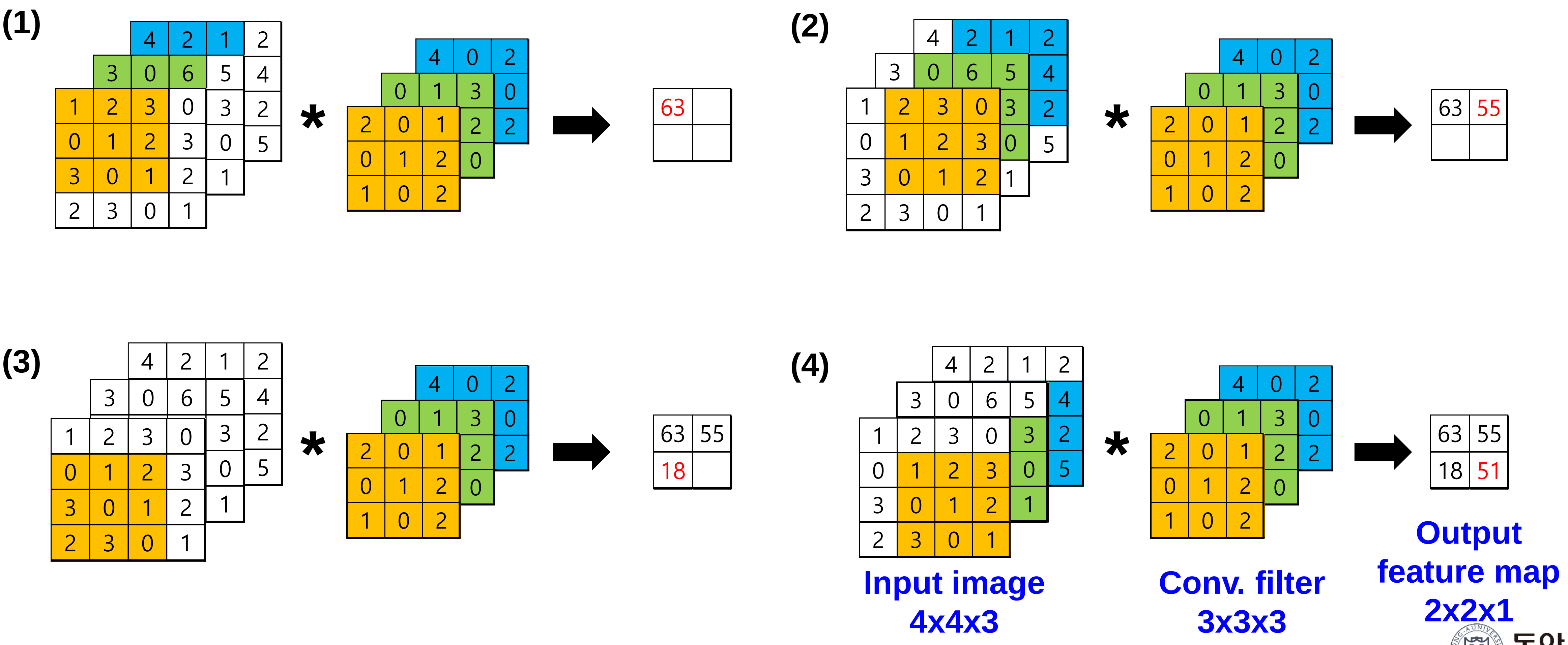
(1)



Input image
4x4x3

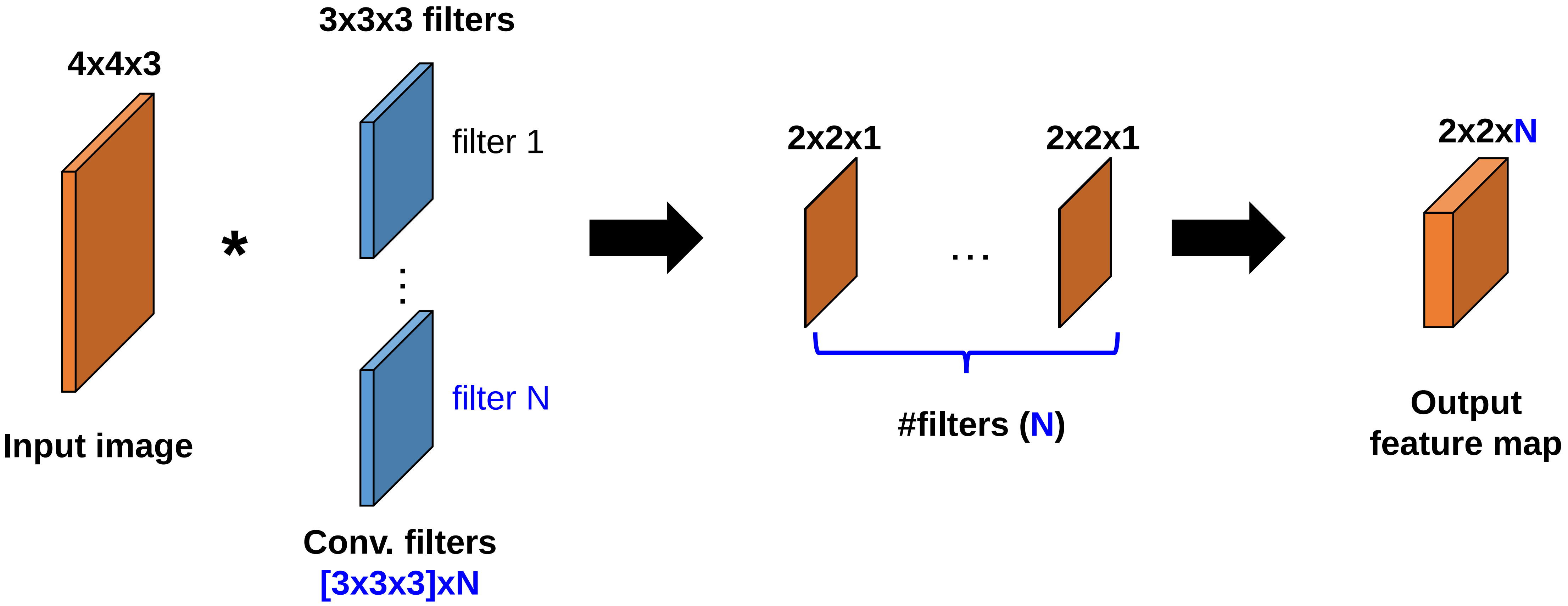
Conv. filter
3x3x3

- 3D 이미지 (RGB) 입력에 대한 2D convolution 연산



Review – Convolutional layer

- **3D 이미지 (RGB) 입력에 대한 2D convolution 연산**
  - Output feature map의 채널 수는 filter의 개수(N)와 같음



Review – Convolutional layer

- **Stride: Convolution 연산의 step size**
  - Stride가 커질수록 feature map의 크기는 작아짐

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

*

2	0	1
0	1	2
1	0	2

→

15				

Stride=1

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

*

2	0	1
0	1	2
1	0	2

→

15	16			

Input image **Filter** **Output feature map**

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

*

2	0	1
0	1	2
1	0	2

→

15		

Stride=2

1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1
2	3	0	1	2	3	0
1	2	3	0	1	2	3
0	1	2	3	0	1	2
3	0	1	2	3	0	1

*

2	0	1
0	1	2
1	0	2

→

15	17	

Input image **Filter** **Output feature map**

Review – Convolutional layer

- **Padding: Input image 주변 값을 특정 값 (주로 0) 으로 채워 줌**
  - Convolution 연산으로 boundary 정보가 소실되는 문제를 방지

Padding size = 0

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

Input image

*****
stride: 1

2	0	1
0	1	2
1	0	2

Filter

➡

15	16
6	15

Output feature map

Padding size = 1

0	0	0	0	0	0
0	1	2	3	0	0
0	0	1	2	3	0
0	3	0	1	2	0
0	2	3	0	1	0
0	0	0	0	0	0

Input image

*****
stride: 1

2	0	1
0	1	2
1	0	2

Filter

➡

7	12	10	2
4	15	16	10
10	6	15	6
8	4	7	3

Output feature map

Review – Convolutional layer

- **Convolution 연산의 출력 크기 계산**
  - 출력의 크기는 filter size, stride, padding size에 따라 달라짐
  - 출력 크기는 정수로 나누어 떨어져야 함

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

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

- (H, W): Input data size
- (FW, FH): Filter size
- P: Padding size
- S: Stride

Implementation of Convolutional Layer

- **torch.nn.Conv2d() 함수를 이용한 합성곱 계층 구현**

CONV2D

CLASS torch.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*) [SOURCE]

① in_channels: 입력 특징맵의 채널 개수

② out_channels: 출력 특징맵의 채널 개수

③ kernel_size: 커널 크기

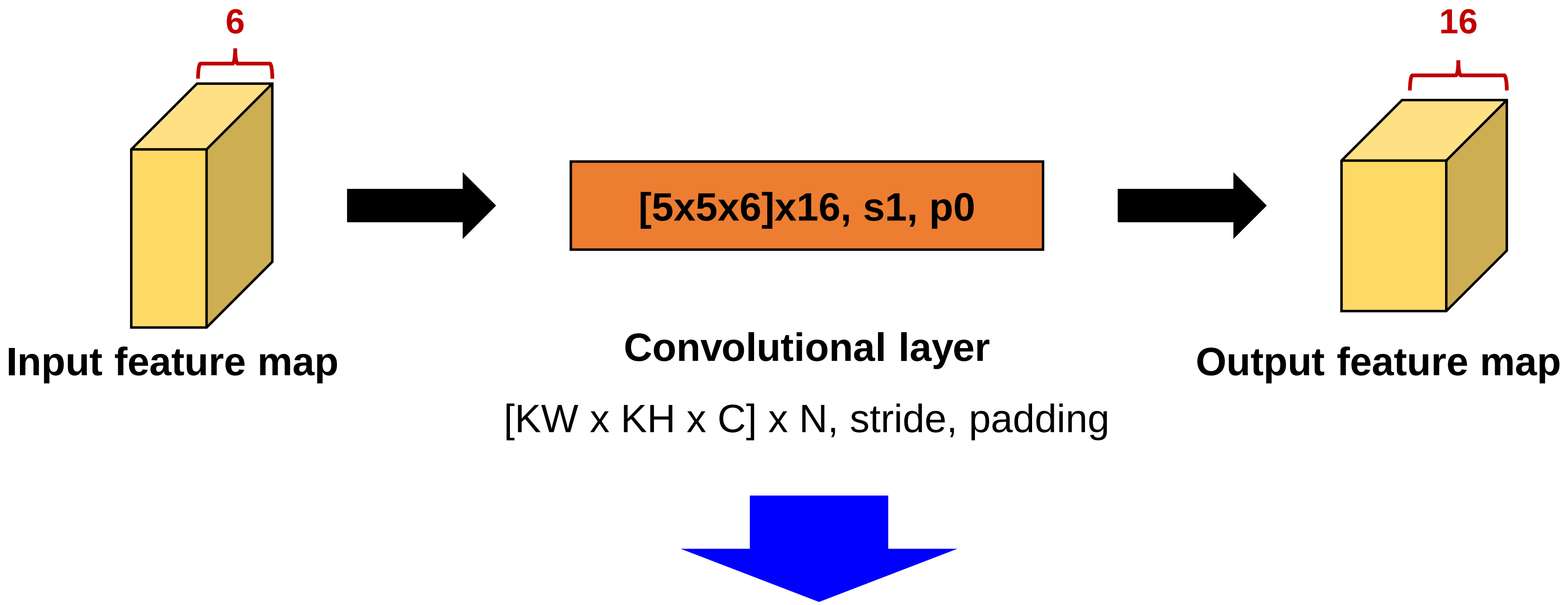
④ stride: stride 크기

⑤ padding: padding 크기

Ref.: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html?highlight=conv2d#torch.nn.Conv2d

Implementation of Convolutional Layer

- **torch.nn.Conv2d() 함수를 이용한 합성곱 계층 구현**

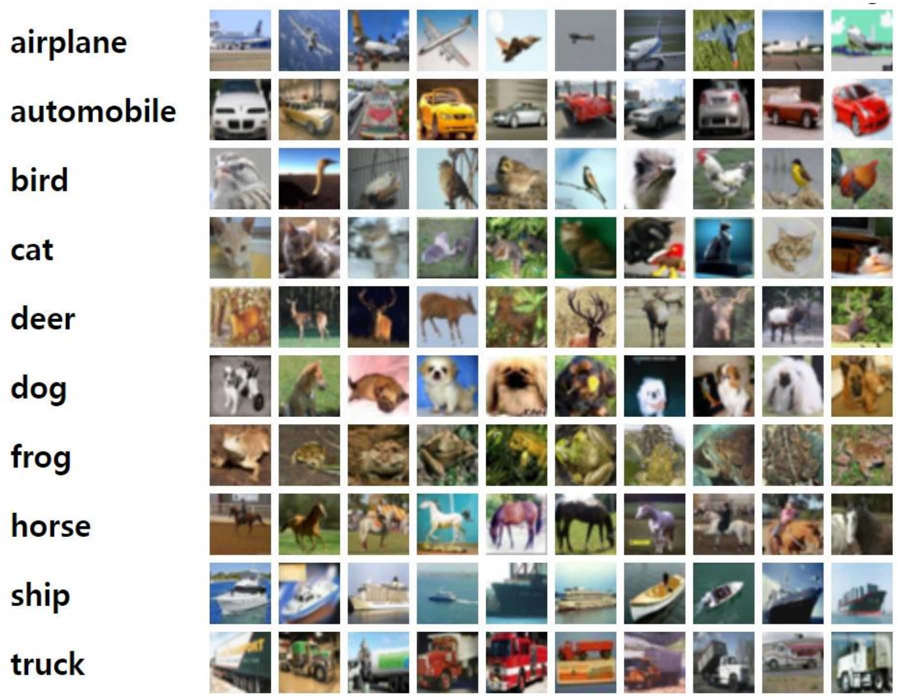


self.conv = nn.Conv2d (in_channels = 6, out_channels = 16, kernel_size = 5, stride = 1, padding = 0)

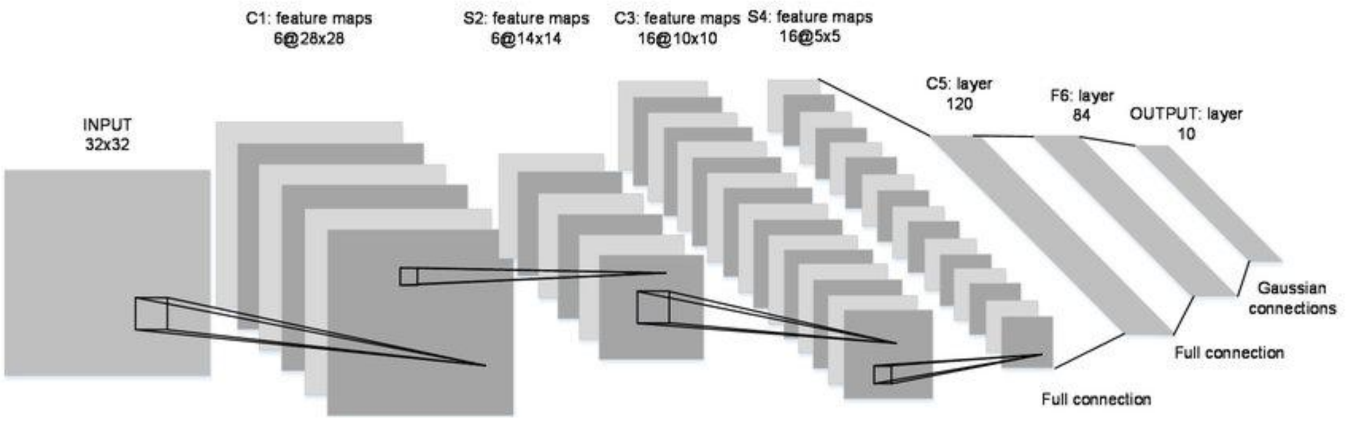
Ref.: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html?highlight=conv2d#torch.nn.Conv2d

CIFAR-10 분류 실습 – CNN을 이용한 분류

- **CIFAR-10 dataset 형상**
  - 32x32x3 (RGB) 이미지, 10개의 클래스
  - Train: 50,000개, Test: 10,000개



CIFAR-10 dataset 예시

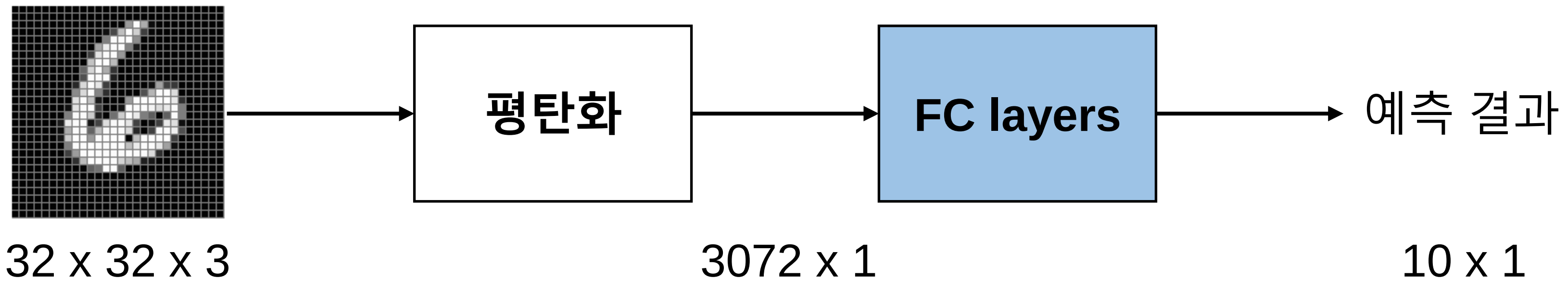


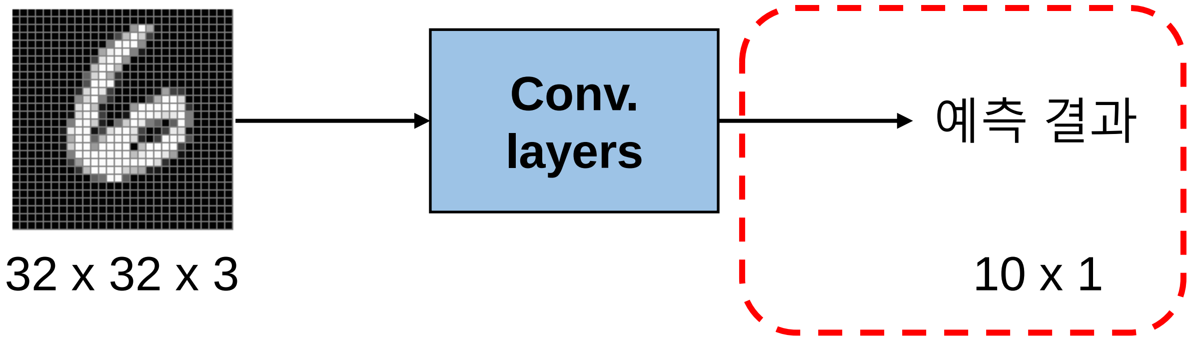
LeNet-5 신경망 구조

동아대학교 DONG-A UNIVERSITY

CIFAR-10 분류 실습 – CNN을 이용한 분류

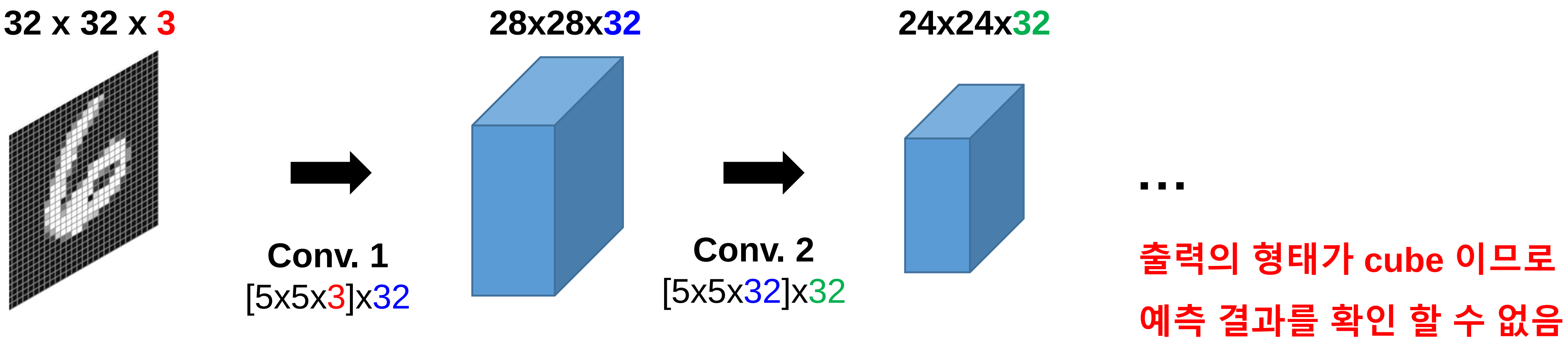
- **입출력 구조 확인**





CIFAR-10 분류 실습 – CNN을 이용한 분류

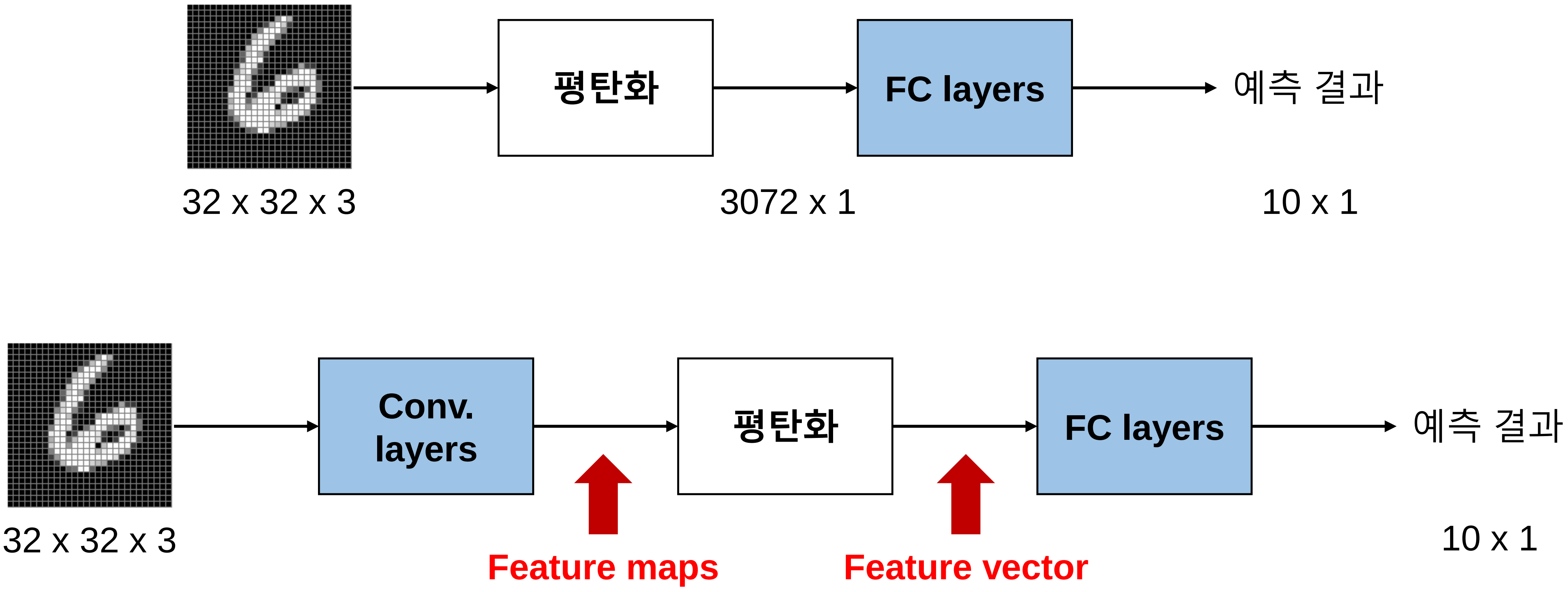
- 입출력 구조 확인



CNN model 예시

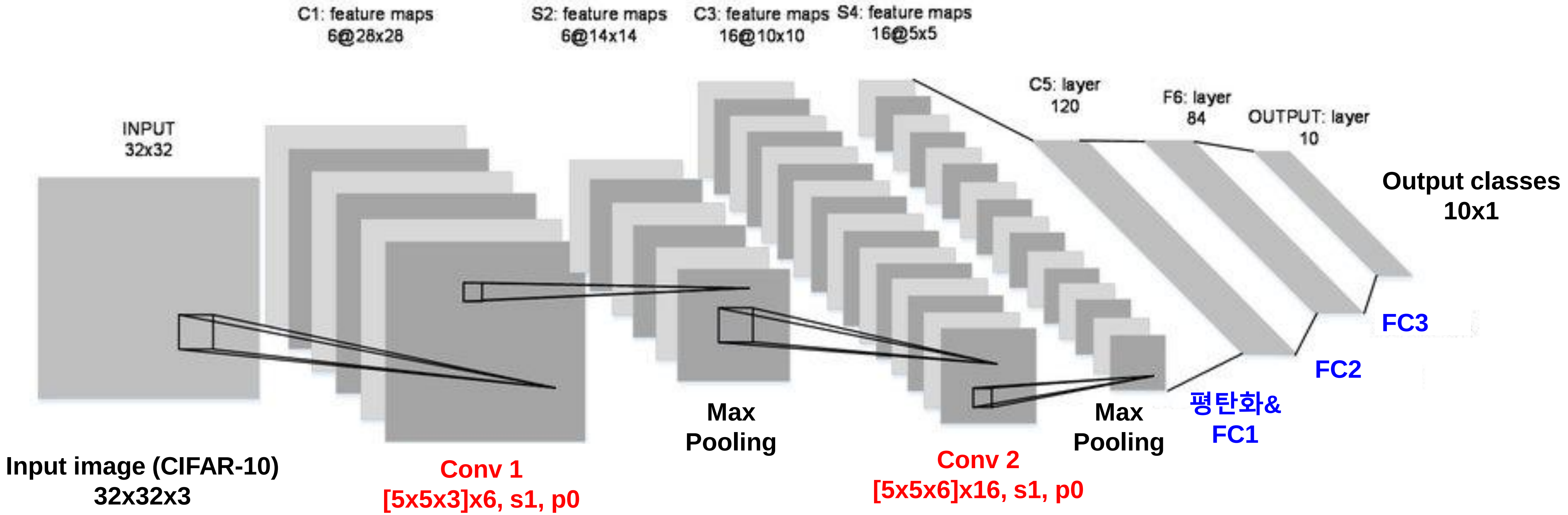
CIFAR-10 분류 실습 – CNN을 이용한 분류

- 입출력 구조 확인



CIFAR-10 분류 실습 – CNN을 이용한 분류

- **LeNet-5 신경망 구조**
  - 2개의 convolutional layer, 3개의 fully connected layer로 구성



CIFAR-10 분류 실습 – CNN을 이용한 분류

▪ **10주차 LMS 강의 콘텐츠에 업로드 되어있는 base code 다운로드**

[1] 패키지 선언

```
[1]  import torch
     import torch.nn as nn
     import torchvision.datasets as dataset
     import torchvision.transforms as transform
     from torch.utils.data import DataLoader
     import numpy as np
```

[2] 데이터셋 다운로드

- 데이터셋 다운로드 전 구글 드라이브 마운트 및 경로 확인 필요

```
[4]  datasetPath = "./drive/MyDrive/dataset/"
```

```
cifar10_train = dataset.CIFAR10(root = datasetPath,
                                train = True,
                                transform = transform.ToTensor(),
                                download = True)
```

[3] Model 구조 선언

- Convolutional layer -> nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)
- Fully connected layer -> nn.Linear(in_features, out_features)
- Max pooling layer -> nn.MaxPool2d(kernel_size, stride)
- Avg pooling layer -> nn.AvgPool2d(kerenel_size, stride)

```
[ ]  class Model(nn.Module):
       def __init__(self):
         super(Model, self).__init__()
       def forward(self, x):
         return x
```

[4] Hyper-parameter 지정

```
[28]  device = 'cuda:0'

      learning_rate = 0.1
      training_epochs = 30
      batch_size = 100
      loss_function = nn.CrossEntropyLoss()
      network = Model().to(device)
```

[5] Training loop

```
[33]  for epoch in range(training_epochs):
        network.train()
        avg_cost = 0
        total_batch = len(data_loader)
```

[6] Test

```
[34]  with torch.no_grad(): # test에서는 기울기 계산 제외
        network.eval()
        img_test = torch.tensor(np.transpose(cifar10_test.data,(0,3,1,2))) / 255.
        label_test = torch.tensor(cifar10_test.targets)

        img_test = img_test.to(device)
        label_test = label_test.to(device)
```

CIFAR-10 분류 실습 – CNN을 이용한 분류

- **10주차 LMS 강의 콘텐츠에 업로드 되어있는 base code 다운로드**

[1] 패키지 선언

```
[1]  import torch
     import torch.nn as nn
     import torchvision.datasets as dataset
     import torchvision.transforms as transform
     from torch.utils.data import DataLoader
     import numpy as np
```

[2] 데이터셋 다운로드

- 데이터셋 다운로드 전 구글 드라이브 마운트 및 경로 확인 필요

```
[4]  datasetPath = "./drive/MyDrive/dataset/"
```

```
cifar10_train = dataset.CIFAR10(root = datasetPath,
                                train = True,
                                transform = transform.ToTensor(),
                                download = True)
```

[3] Model 구조 선언

- Convolutional layer -> nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)
- Fully connected layer -> nn.Linear(in_features, out_features)
- Max pooling layer -> nn.MaxPool2d(kernel_size, stride)
- Avg pooling layer -> nn.AvgPool2d(kerenel_size, stride)

```
[ ]  class Model(nn.Module):
       def __init__(self):
         super(Model, self).__init__()
       def forward(self, x):
         return x
```

LeNet5 모델 구조에 맞추어 코드 작성

[4] Hyper-parameter 지정

```
[28]  device = 'cuda:0'

      learning_rate = 0.1
      training_epochs = 30
      batch_size = 100
      loss_function = nn.CrossEntropyLoss()
      network = Model().to(device)
```

[5] Training loop

```
[33]  for epoch in range(training_epochs):
        network.train()
        avg_cost = 0
        total_batch = len(data_loader)
```

[6] Test

```
[34]  with torch.no_grad(): # test에서는 기울기 계산 제외
        network.eval()
        img_test = torch.tensor(np.transpose(cifar10_test.data,(0,3,1,2))) / 255.
        label_test = torch.tensor(cifar10_test.targets)

        img_test = img_test.to(device)
        label_test = label_test.to(device)
```

동아대학교
DONG-A UNIVERSITY

CIFAR-10 분류 실습 – CNN을 이용한 분류

- **LeNet-5 모델 구조 작성 참고사항**
  - 참고자료: https://pytorch.org/docs/stable/nn.html

CONV2D

CLASS `torch.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)` [SOURCE]

RELU

CLASS `torch.nn.ReLU(inplace=False)` [SOURCE]

MAXPOOL2D

CLASS `torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_i ceil_mode=False)` [SOURCE]

LINEAR

CLASS `torch.nn.Linear(in_features, out_features, bias=True, device=None, dtype=None)` [SOURCE]

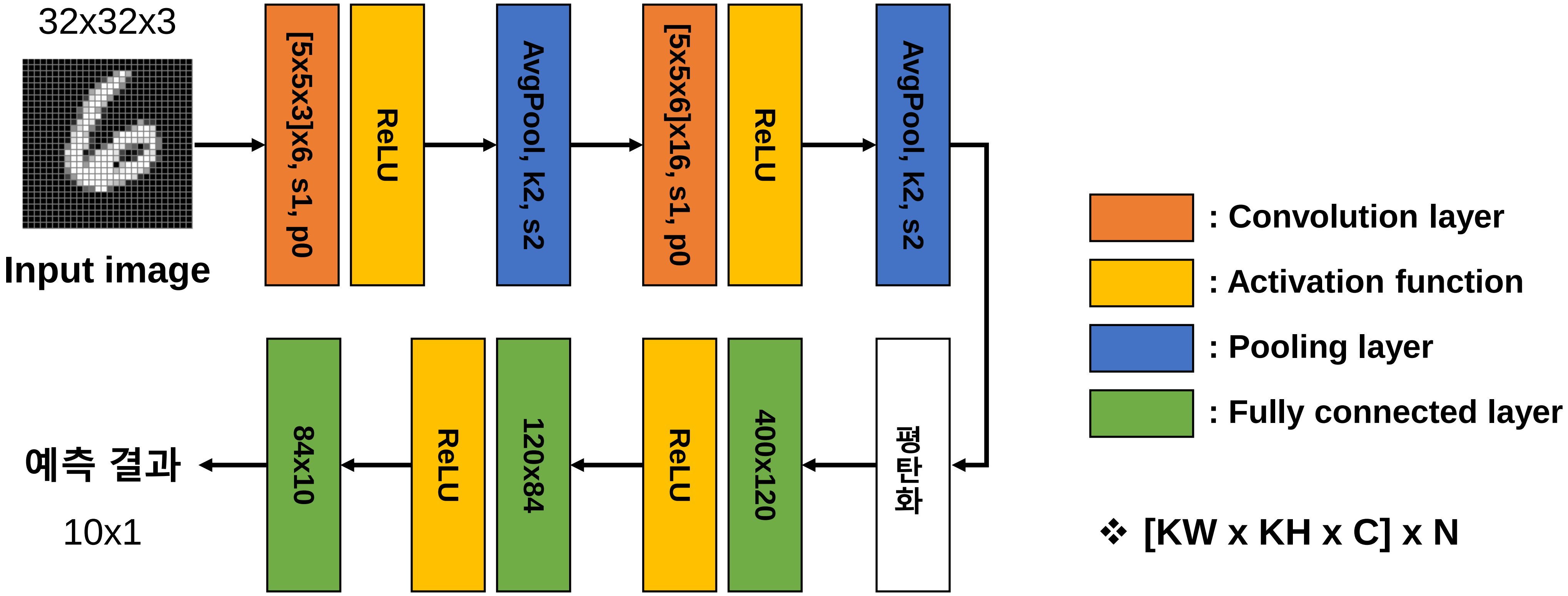
: Convolution layer

: Activation function

: Pooling layer

: Fully connected layer

- **LeNet-5 모델 구조 작성 참고사항**
  - Filter size: 5x5, Stride: 1, Padding: 0



LeNet-5 구조

CIFAR-10 분류 실습 – CNN을 이용한 분류

- **Convolution 연산의 출력 크기 계산**
  - 출력의 크기는 filter size, stride, padding size에 따라 달라짐
  - 출력 크기는 정수로 나누어 떨어져야 함

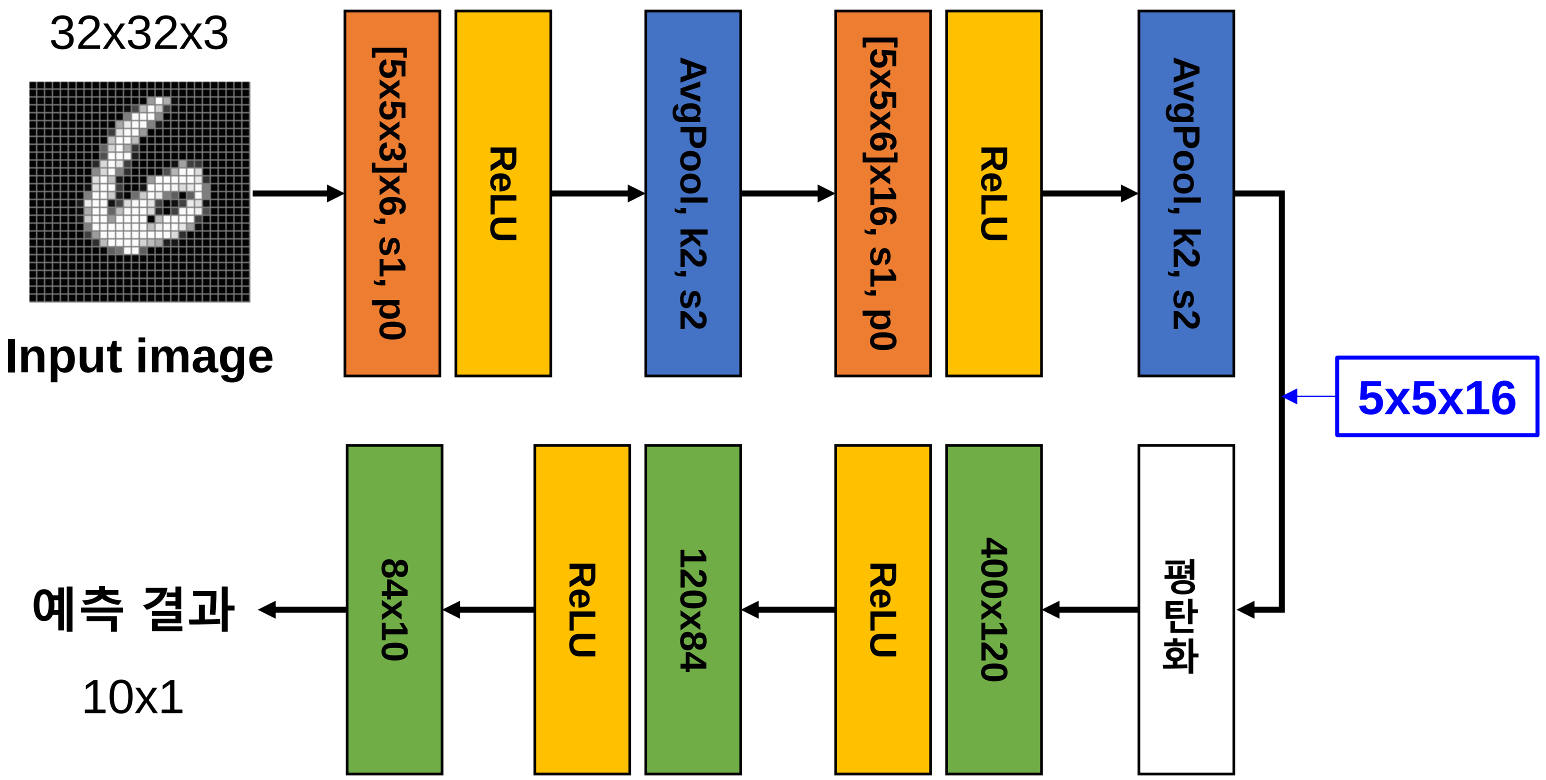
$$\text{Output Height} = OH = \frac{H + 2P - FH}{S} + 1$$

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

- (H, W): Input data size
- (FW, FH): Filter size
- P: Padding size
- S: Stride

CIFAR-10 분류 실습 – CNN을 이용한 분류

- **LeNet-5 모델 구조 작성 참고사항**
  - Filter size: 5x5, Stride: 1, Padding: 0



LeNet-5 구조

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()

    def forward(self, x):

        # convolutional layers

        y = torch.reshape(y, (-1, 5*5*16))

        # fully connected layers

        return y
```

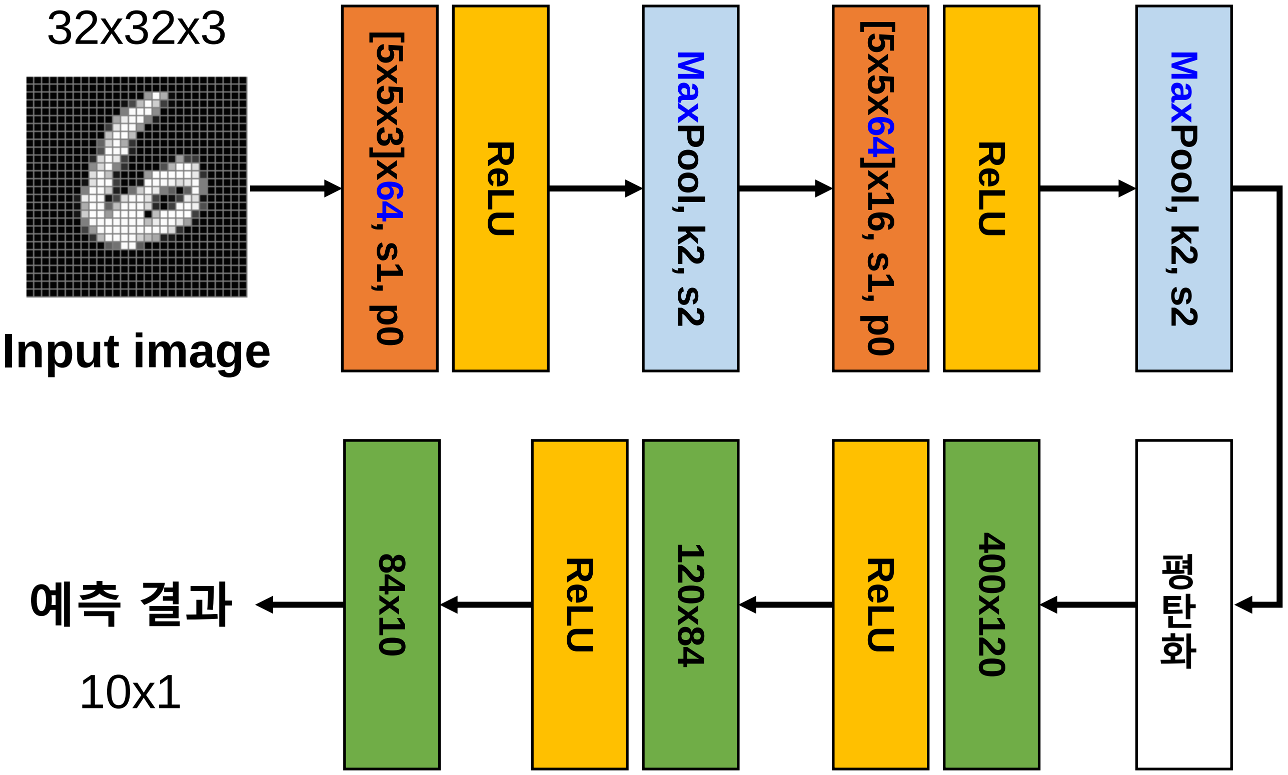
특징맵 평탄화

5x5x16 → 400x1

동아대학교 DONG-A UNIVERSITY

Appendix – 더 높은 정확도를 가지는 LeNet-5 설계

1. **Pooling layer 변경: Average pooling → Max pooling**
2. **Convolutional layer channel 개수 변경: 6 → 32, 64**



LeNet-5 구조

Appendix – 더 높은 정확도를 가지는 LeNet-5 설계

3. Learning rate control

- 1 ~ 74 epoch: 0.001
- 75 ~ 149 epoch: 0.0005
- 150 ~ 200 epoch: 0.00025

```
# hyper-parameter 변경
training_epochs = 200
scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=[75, 150], gamma=0.5)
```

[6] Training loop

```
[20] for epoch in range(training_epochs):
         avg_cost = 0
         total_batch = len(data_loader)

         for img, label in data_loader:
             pred = network(img)

             loss = loss_function(pred, label)
             optimizer.zero_grad()
             loss.backward()
             optimizer.step()
             avg_cost += loss / total_batch

         print('Epoch: %d, LR: %f,  Loss: %f' %(epoch+1, optimizer.param_groups[0]['lr'], avg_cost))
         scheduler.step()

     print('Learning finished')
```

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Dept. of Computer Engineering
Dong-A University, Busan, Rep. of Korea