

컨볼루션 연산과 영상 분류 CNN 모델

이건명

충북대학교 소프트웨어학과

학습 내용

- 다양한 컨볼루션 연산에 대해서 알아본다.
- 물체인식 CNN 모델들에 대해서 알아본다.

1. 컨볼루션의 형태

- 단일 채널 컨볼루션
- 다중 채널 2D 컨볼루션
- 다중 채널 3D 컨볼루션
- 1x1 컨볼루션
- 디컨볼루션
- 팽창 컨볼루션
- 공간 분할 컨볼루션
- 깊이별 분할 컨볼루션
- 집단 컨볼루션
- 채널섞기 집단 컨볼루션

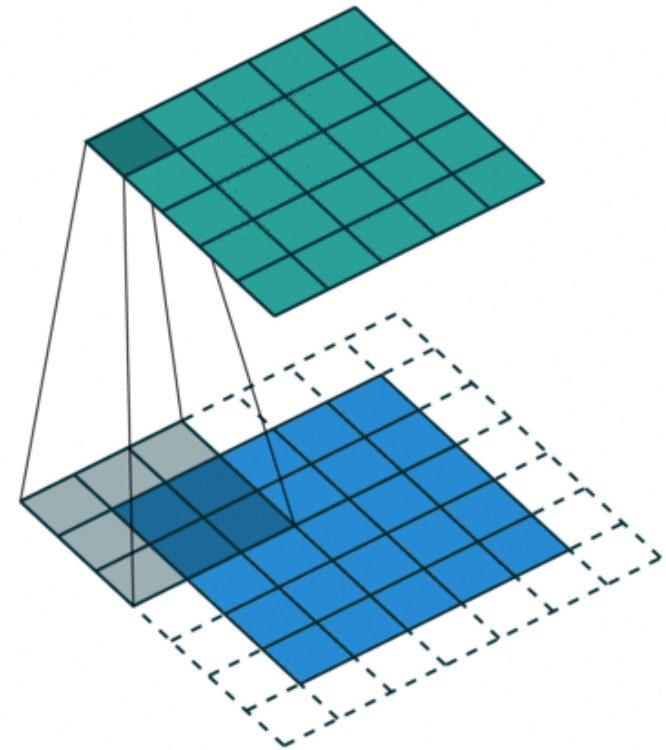
컨볼루션

❖ 단일 채널 대상의 컨볼루션

- 컨볼루션 커널, 스트라이드(stride), 패딩(padding)

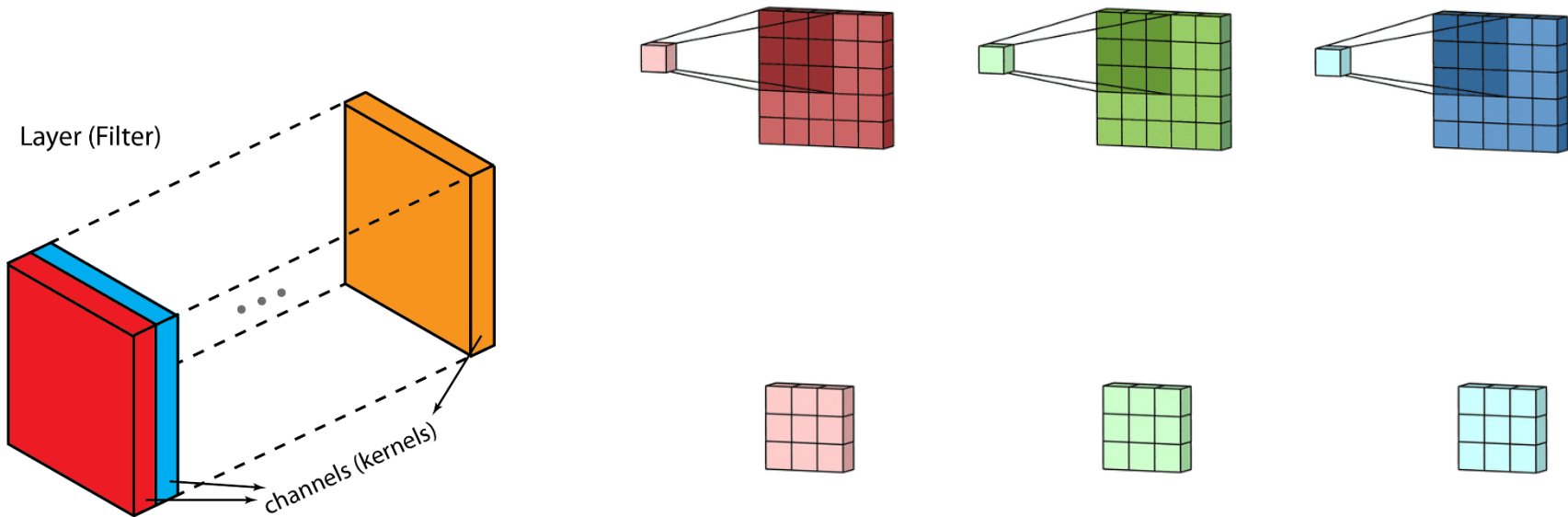
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0



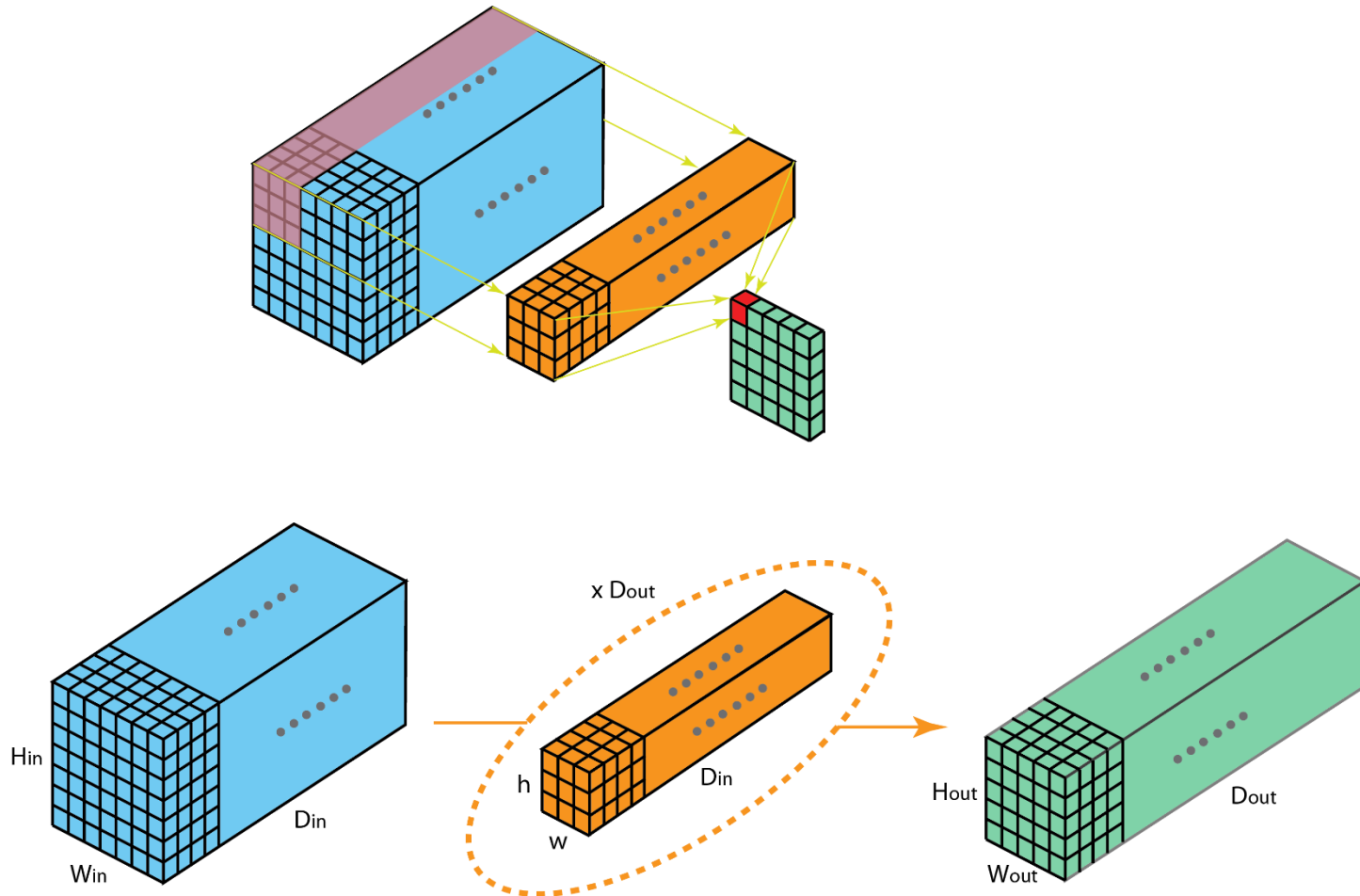
컨볼루션

❖ 다중 채널의 2D 컨볼루션



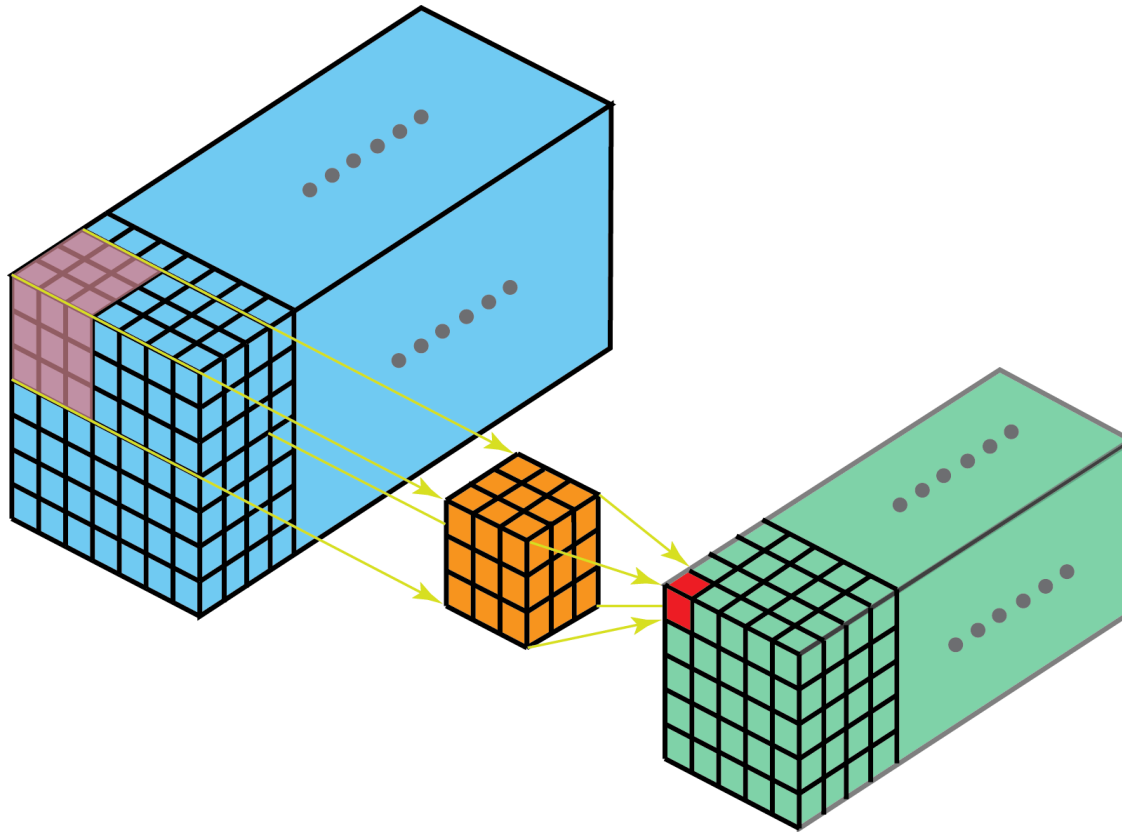
컨볼루션

❖ 다중 채널의 2D 컨볼루션 - cont.



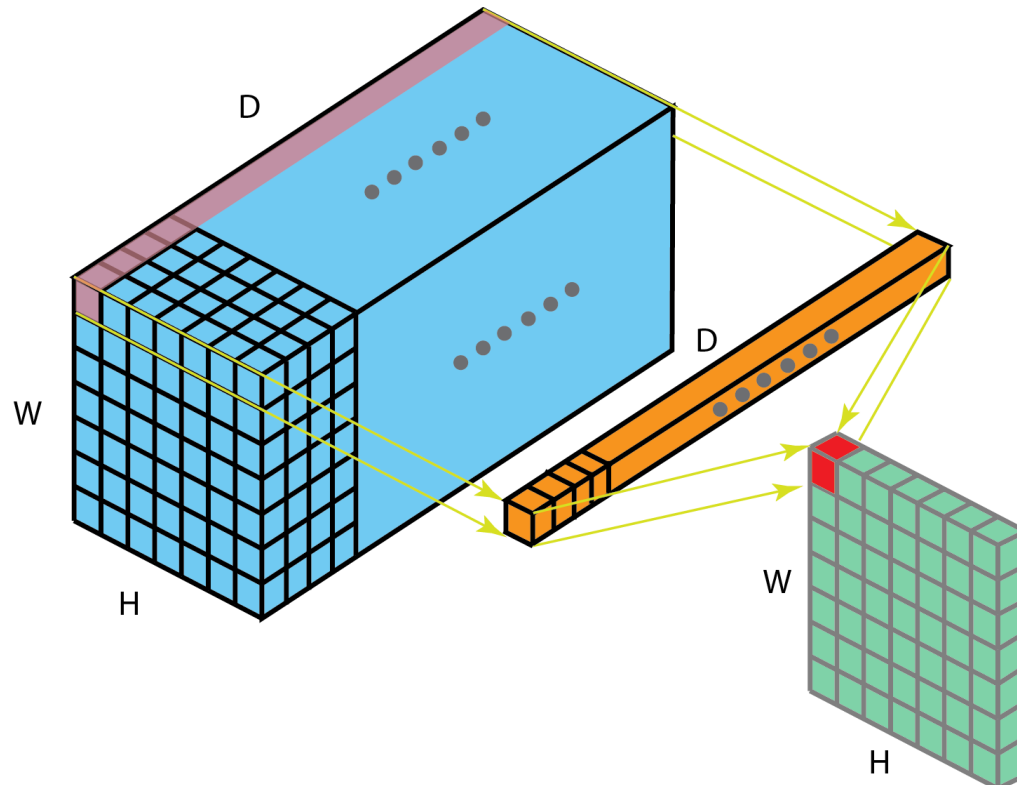
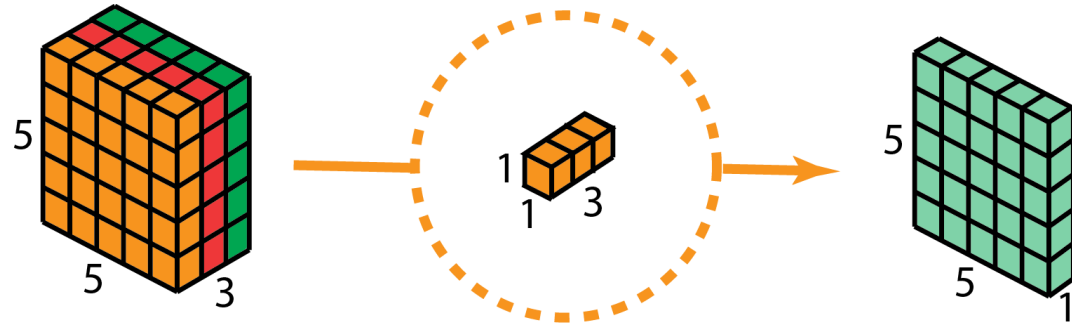
컨볼루션

❖ 3D 컨볼루션



컨볼루션

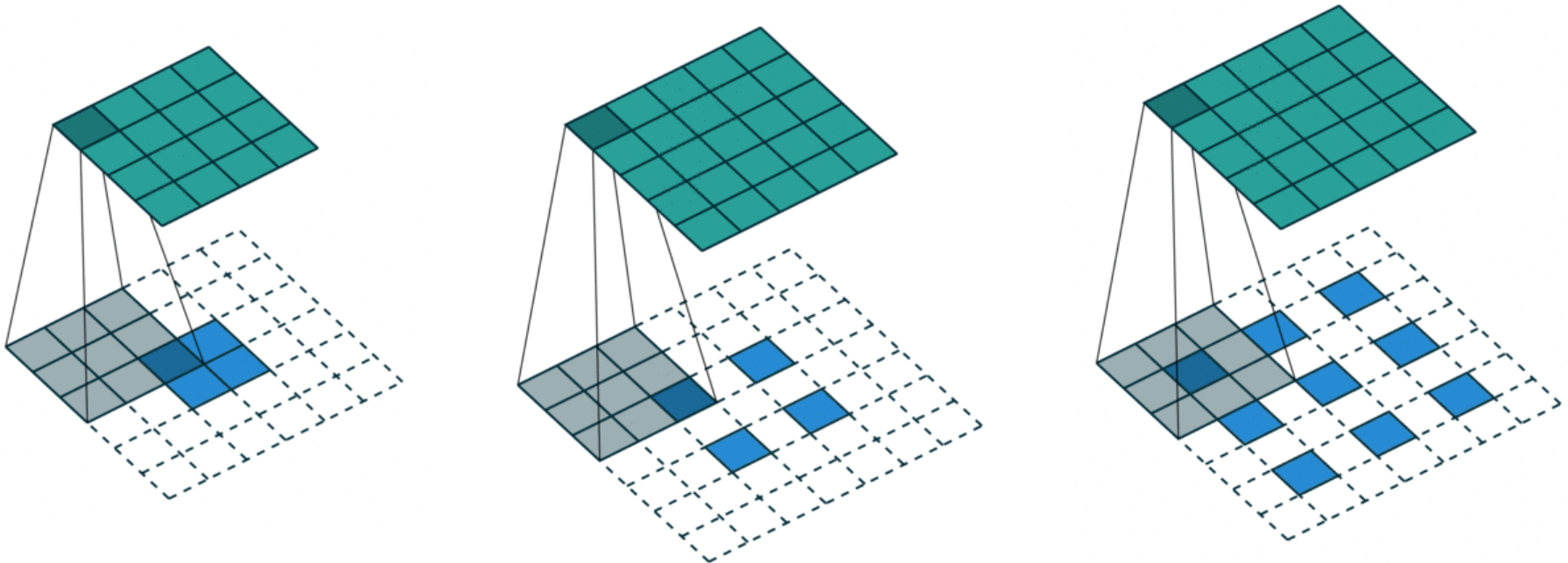
❖ 1 x 1 컨볼루션



컨볼루션

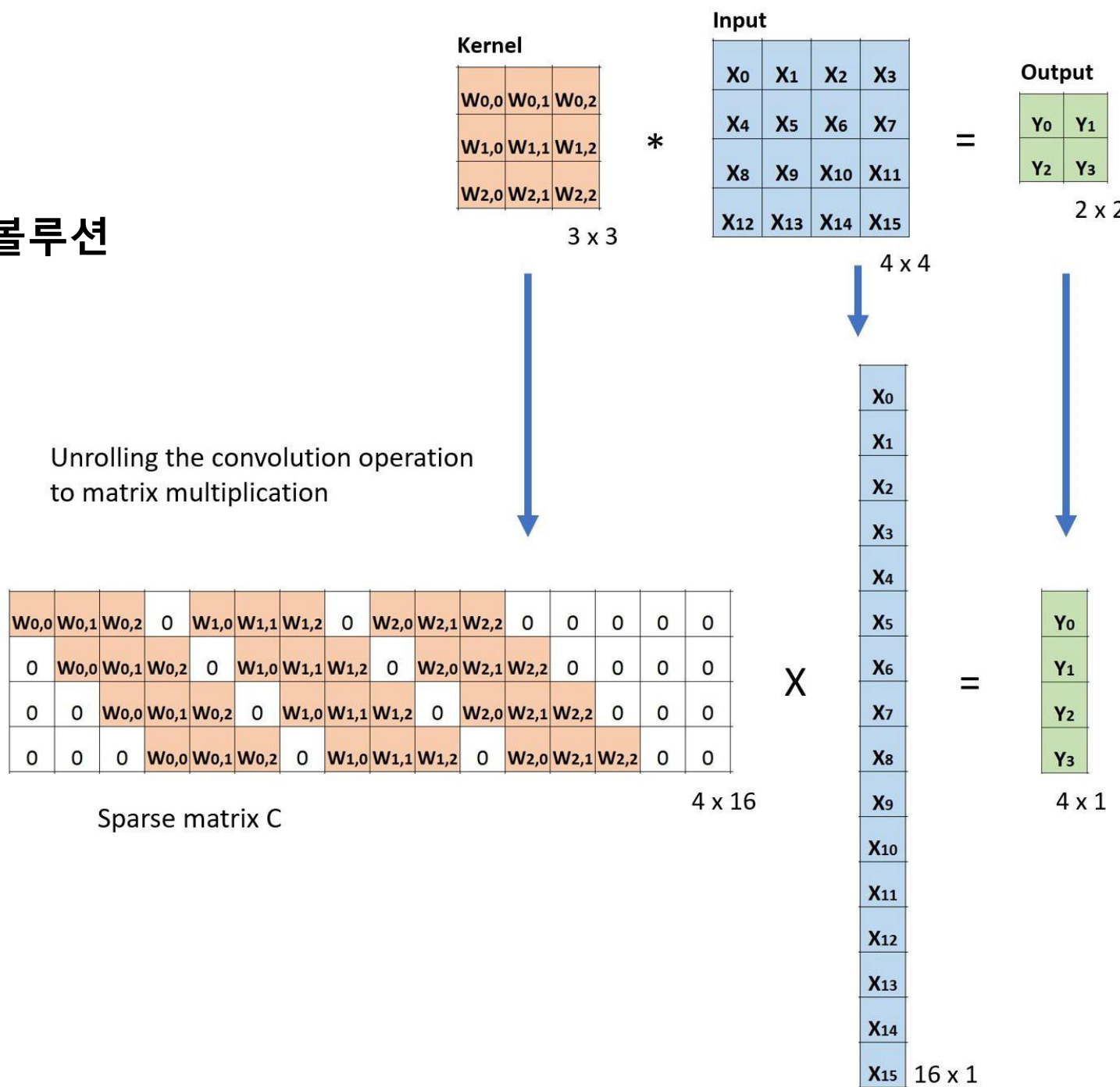
❖ 디컨볼루션(deconvolution, transposed convolution, fractionally-strided convolution)

- 입력보다 큰 출력 생성(upsampling)



컨볼루션

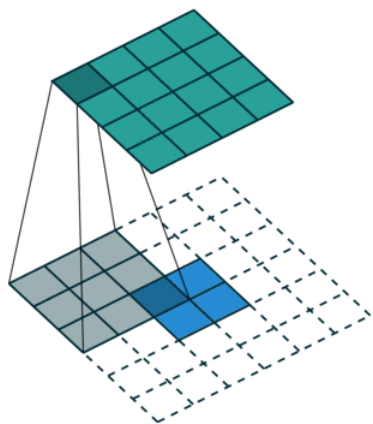
❖ 일반적인 컨볼루션



컨볼루션

transposed convolution 이름의 유래

❖ 디컨볼루션



W _{0,0}	0	0	0
W _{0,1}	W _{0,0}	0	0
W _{0,2}	W _{0,1}	W _{0,0}	0
0	W _{0,2}	W _{0,1}	W _{0,0}
W _{1,0}	0	W _{0,2}	W _{0,1}
W _{1,1}	W _{1,0}	0	W _{0,2}
W _{1,2}	W _{1,1}	W _{1,0}	0
0	W _{1,2}	W _{1,1}	W _{1,0}
W _{2,0}	0	W _{1,2}	W _{1,1}
W _{2,1}	W _{2,0}	0	W _{1,2}
W _{2,2}	W _{2,1}	W _{2,0}	0
0	W _{2,2}	W _{2,1}	W _{2,0}
0	0	W _{2,2}	W _{2,1}
0	0	0	W _{2,2}
0	0	0	0
0	0	0	0

16 x 4

Sparse matrix C^T

\times

Y ₀
Y ₁
Y ₂
Y ₃

4 x 1

=

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

16 x 1

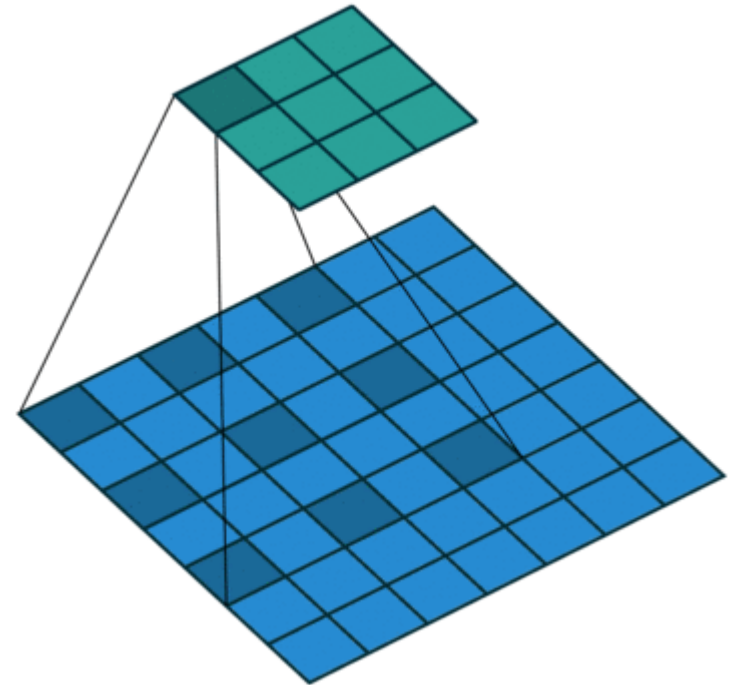
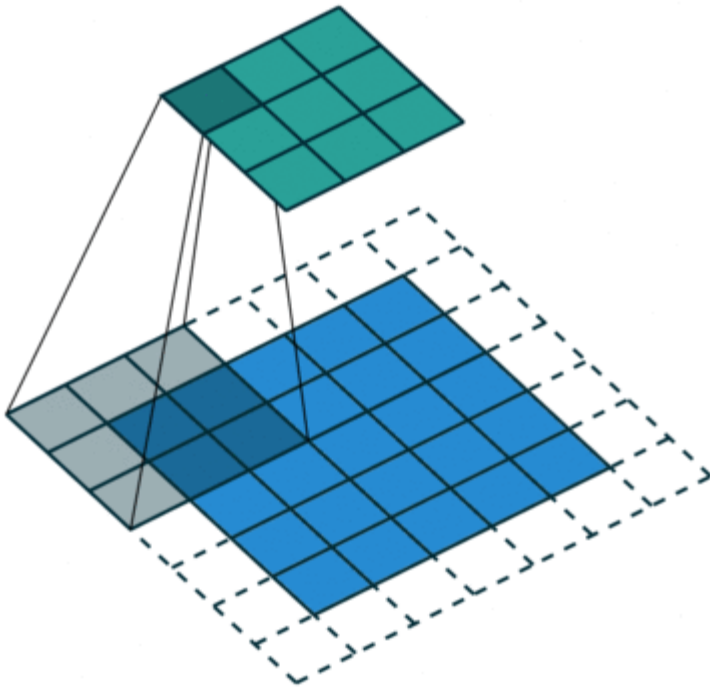


X ₀	X ₁	X ₂	X ₃
X ₄	X ₅	X ₆	X ₇
X ₈	X ₉	X ₁₀	X ₁₁
X ₁₂	X ₁₃	X ₁₄	X ₁₅

4 x 4

컨볼루션

❖ 팽창 컨볼루션(dilated convolution, atrous convolution)

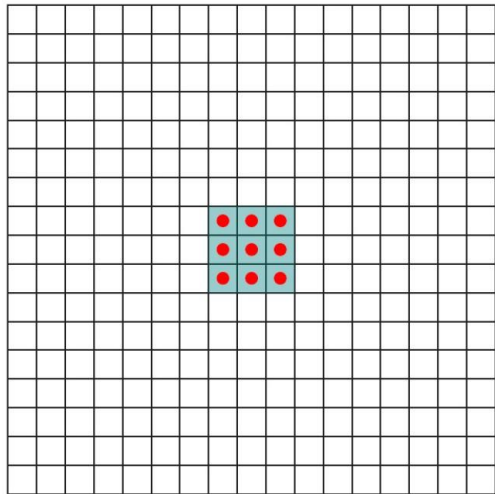


a trous (with holes)

컨볼루션

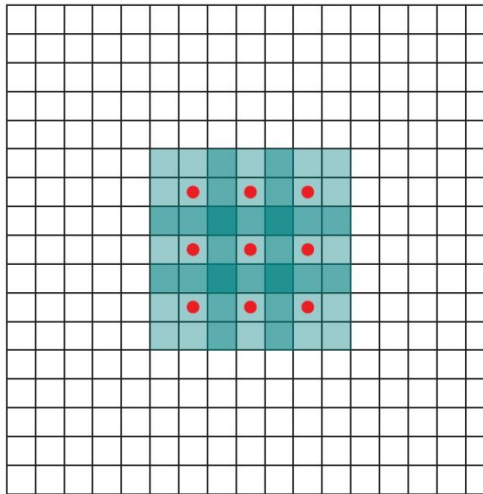
❖ 팽창 컨볼루션(dilated convolution, atrous convolution) – cont.

1 Dilated Convolution



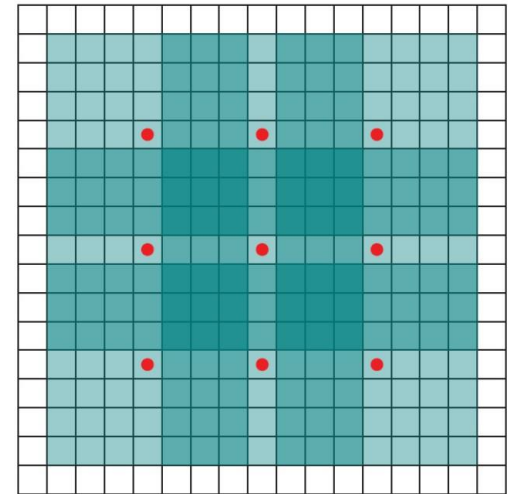
(a)

2 Dilated Convolution



(b)

4 Dilated Convolution



(c)

컨볼루션

❖ 분할(separable) 컨볼루션

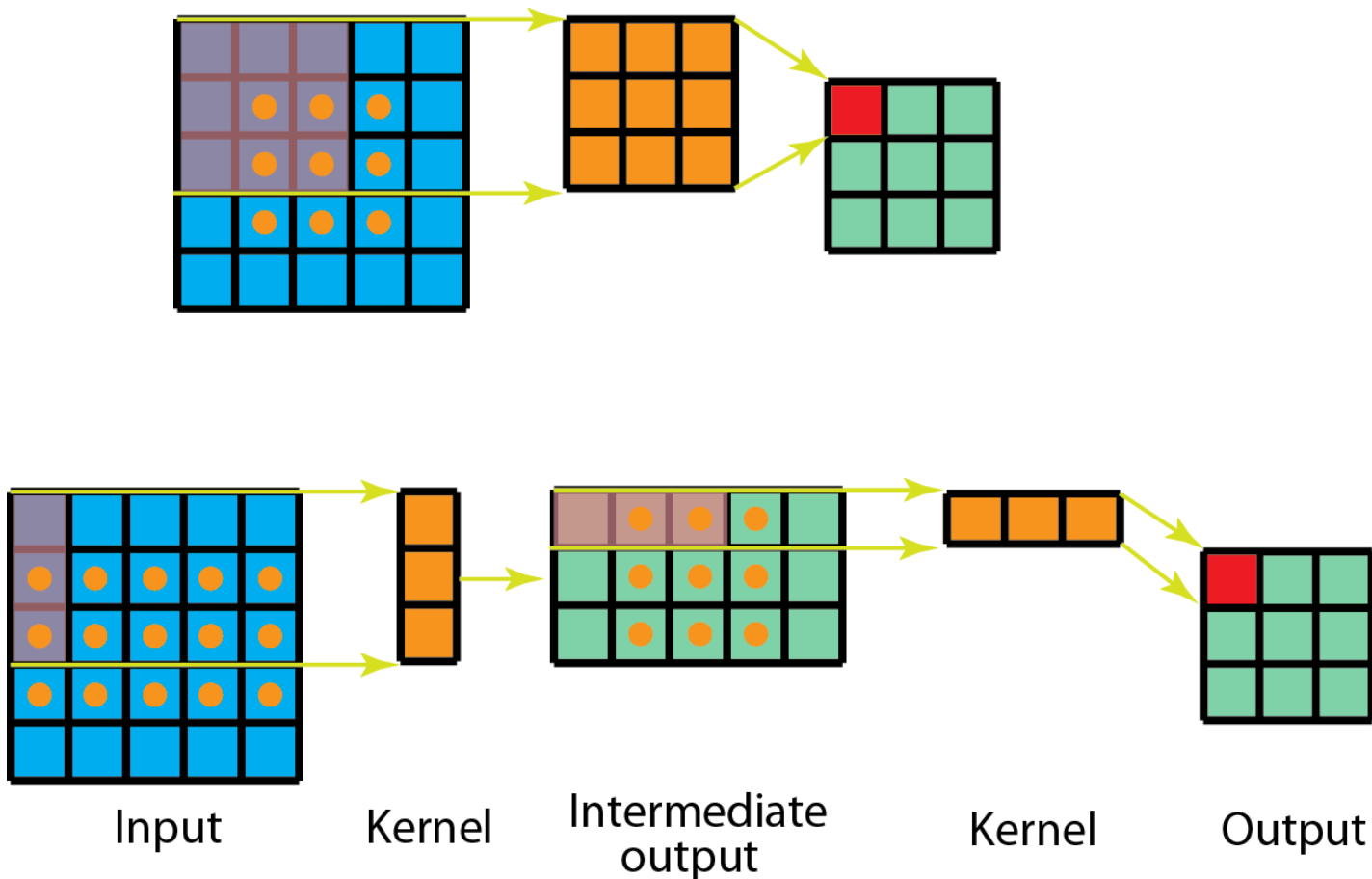
- 공간 분할 컨볼루션
- 깊이별 분할 컨볼루션

❖ 공간 분할 컨볼루션(Spatially Separable Convolutions)

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

컨볼루션

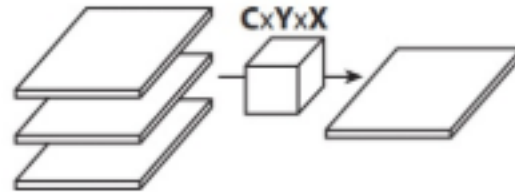
❖ 공간 분할 컨볼루션(Spatially Separable Convolutions)



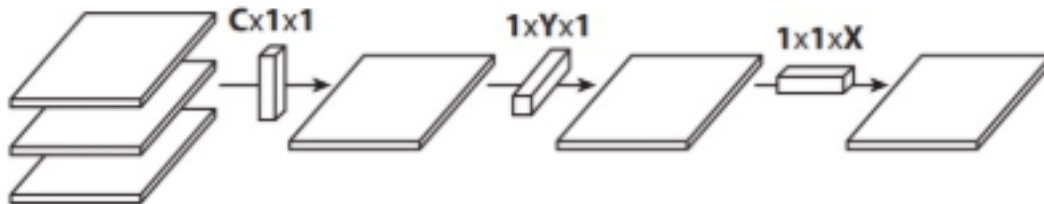
컨볼루션

❖ 공간 분할 컨볼루션

- 평탄화 컨볼루션(flattened convolution)



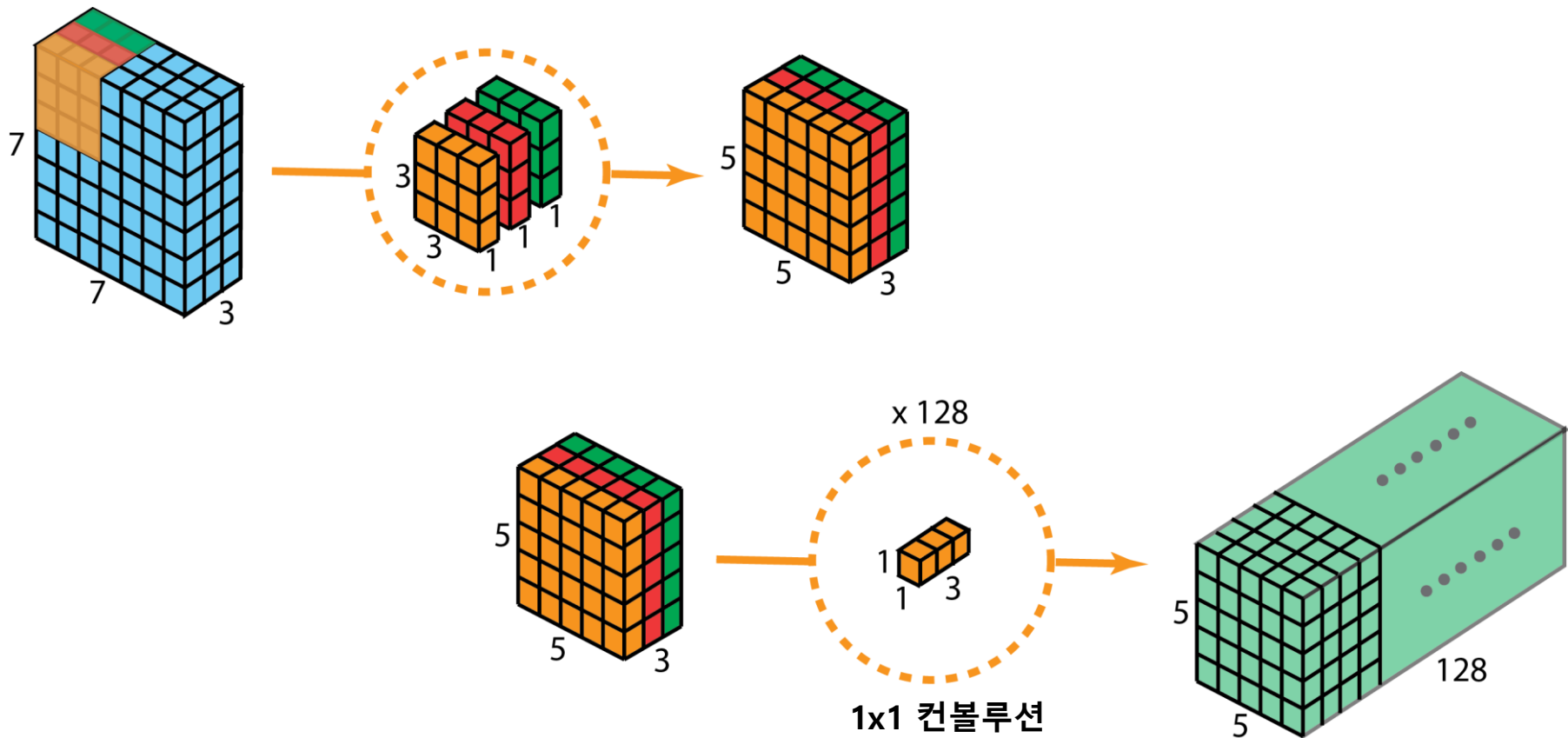
3D 컨볼루션



다른 방향의 1D 컨볼루션

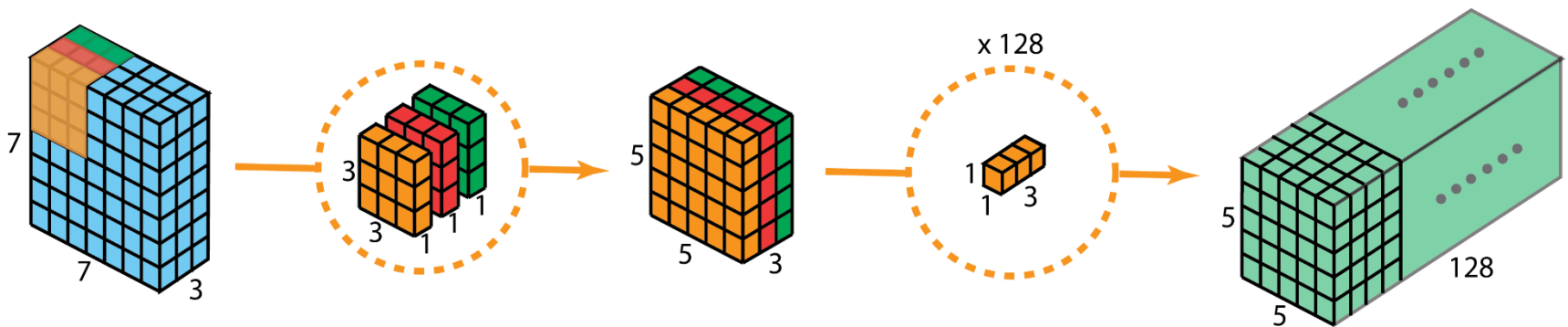
컨볼루션

❖ 깊이별 분할 컨볼루션(Depthwise separable convolution)



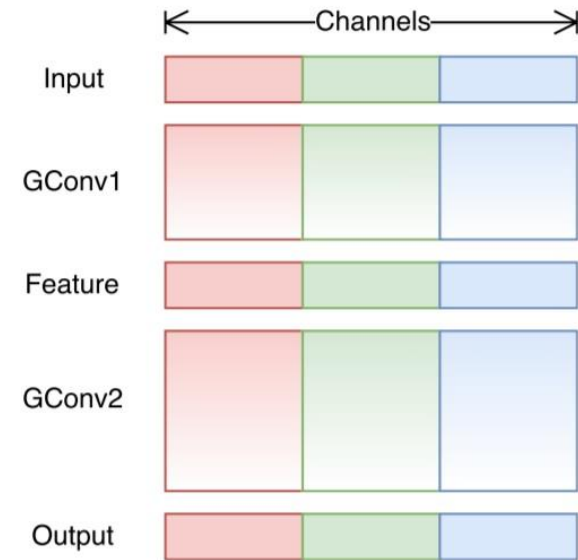
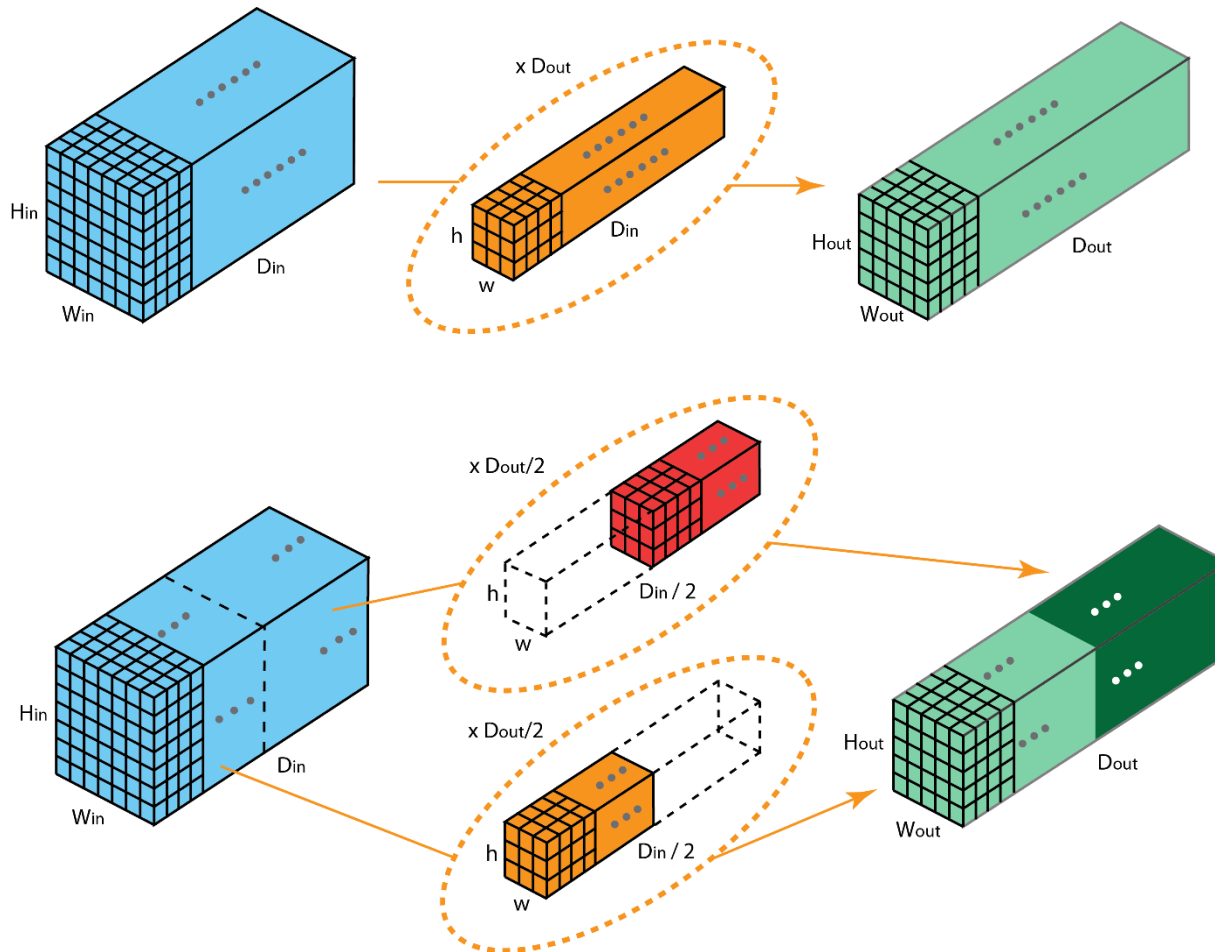
컨볼루션

❖ 깊이별 분할 컨볼루션(Depthwise separable convolution) – cont.



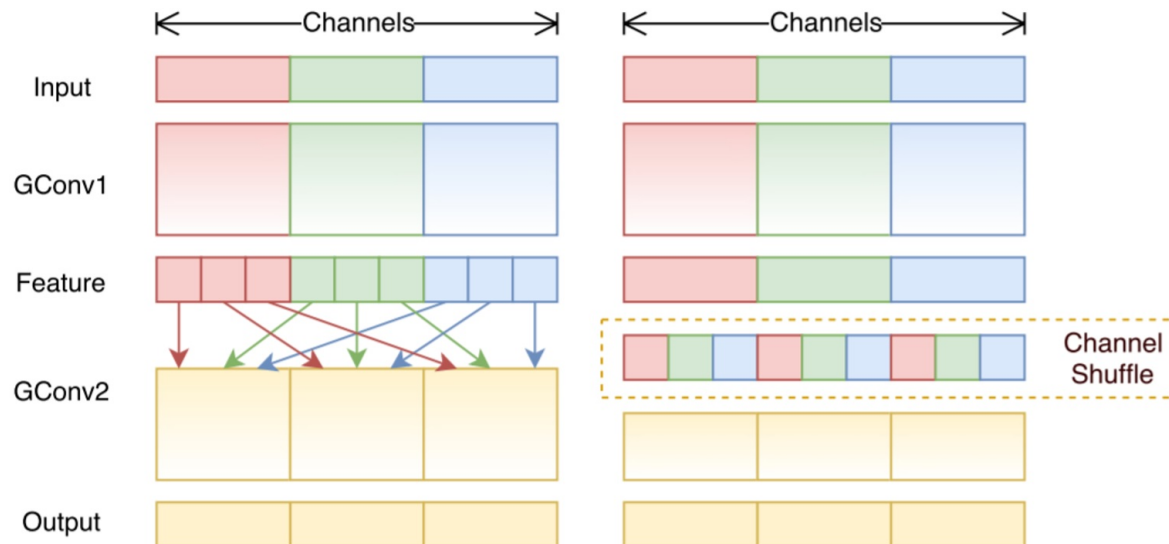
컨볼루션

❖ 집단 컨볼루션(grouped convolution)



컨볼루션

❖ 채널 섞기(channel shuffle)와 채널섞기 집단 컨볼루션(Shuffled Grouped Convolution)



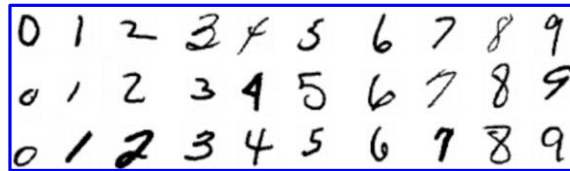
2. 물체 인식 CNN 모델

- LeNet
- AlexNet
- VGGNet
- GoogleNet
- ResNet
- ResNeXt
- DenseNet
- DPN (Dual Path Network)
- SENet
- MobileNet
- SuffleNet

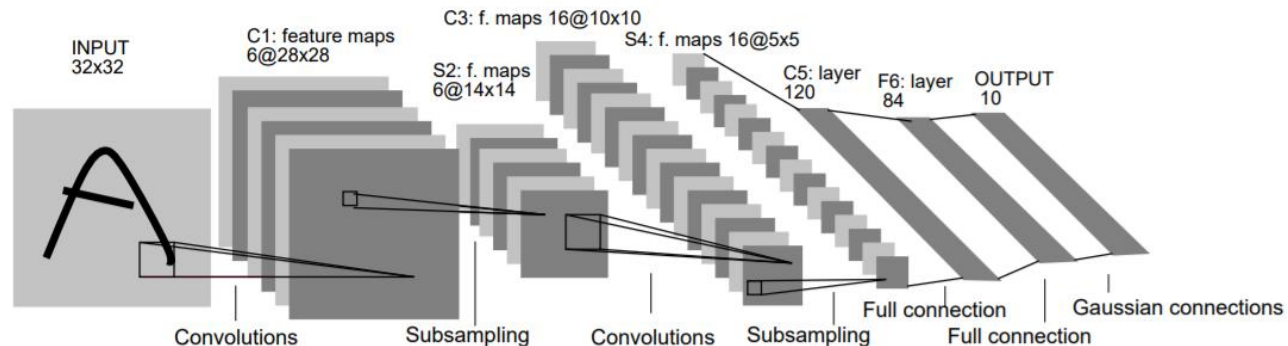
LeNet 모델

❖ LeNet 모델

- Yann LeCun 등의 제안(1998)
- **LeNet5 모델**
 - 5 계층 구조: Conv-Pool-Conv- Pool-Conv-FC-FC(SM)
- 입력 : 32x32 필기체 숫자 영상 (**MNIST** 데이터)



- 풀링 : 가중치x(2x2블록의 합) + 편차항
- 시그모이드 활성화 함수 사용
- 성능: 오차율 0.95%(정확도: 99.05%)



ILSVRC 대회

❖ ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회

- ImageNet 데이터베이스

- 영어 단어 개념의 계층구조인 WordNet에 따라 정리된 영상 데이터베이스



■ 분류 경쟁 부분

- 2010년 시작
- 1,000개의 부류
- 1,200,000 개의 영상 데이터

- 상위-5 오류(top-5 error rate) 평가

ILSVRC 대회

❖ ILSVRC 대회 주요 우수팀

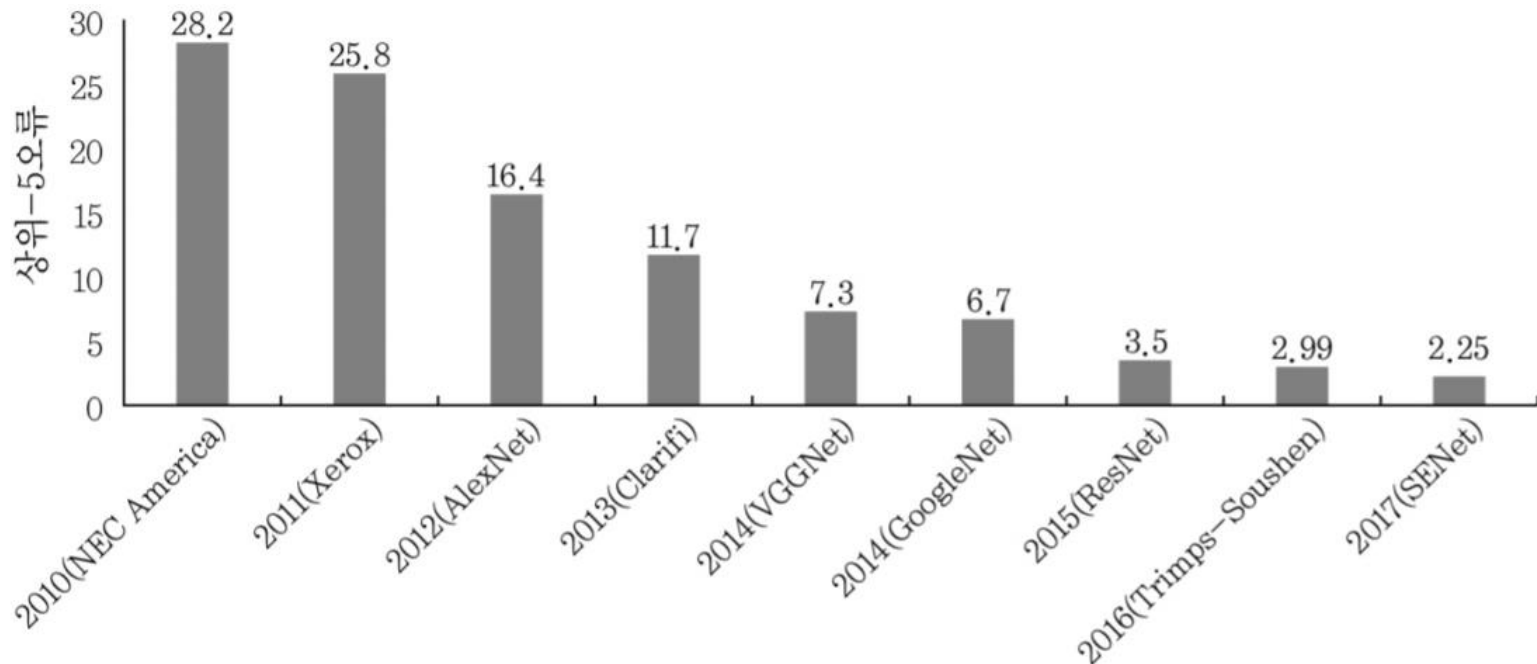


그림 5.19 ILSVRC 주요 우수팀의 성적

가로축은 연도, 괄호 안에는 팀 이름이나 모델 이름을 나타냄

AlexNet 모델

❖ AlexNet

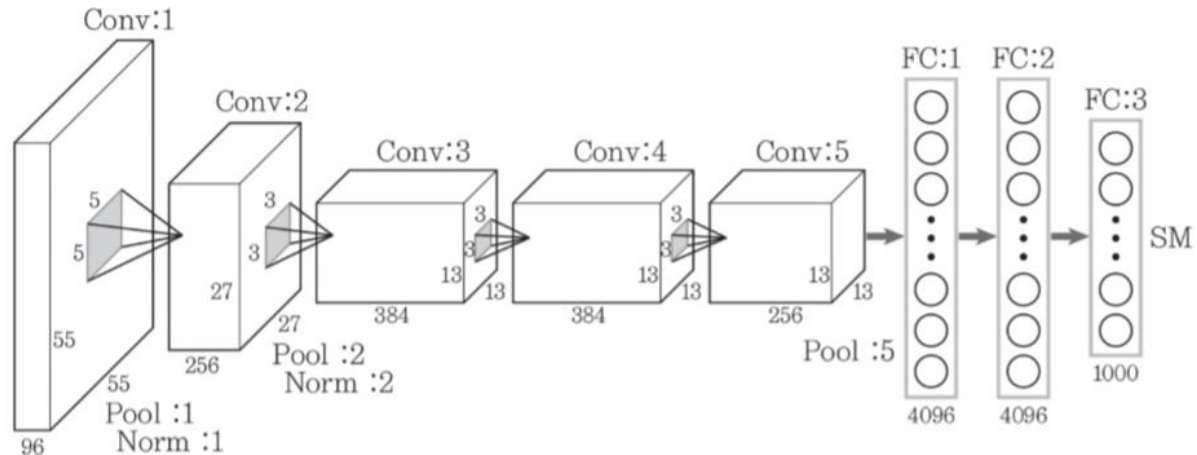
- 토론토 대학 Geoffrey E. Hinton 팀이 제안
- ILSVRC에서 2012년 우승
- 상위-5 오류율 : **16.43%**
 - 직전 년도 대비 9.4% 정확도 향상



AlexNet 모델

❖ AlexNet – cont.

- 8 계층의 구조
 - Conv-Pool-Norm-Conv-Pool-Norm-Conv- Conv-Conv-Pool-FC-FC-FC(SM)




- ReLU 함수를 사용한 첫 모델
- FC 층에 드롭아웃(dropout) 기법 사용
- 최대값 풀링(max pooling) 사용

AlexNet 모델

❖ AlexNet – cont.

- Norm: 국소 반응 정규화 연산 층
 - 인접한 여러 층의 출력값들을 이용하여 출력값 조정

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$


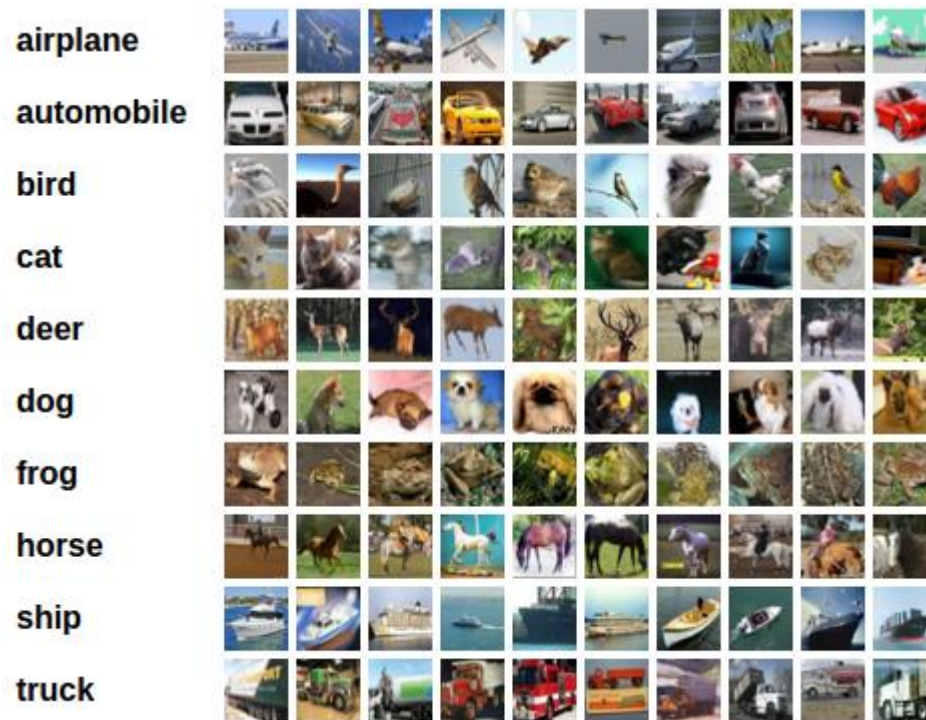
위치 (x, y) 에 커널 i 를 적용하여 계산한 값

- 마지막층
 - 완전연결층(FC층)
 - 소프트맥스(SM) 사용
 - 1,000개의 부류를 나타내기 위해 1,000개의 노드

[실습] CIFAR10 데이터의 인식

❖ CIFAR 10 데이터

- 3 채널의 32x32 크기 10종의 이미지 데이터
- 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'



❖ Colab에서 PyTorch 사용 실습

```
%matplotlib inline
import torch
import torchvision
import torchvision.transforms as transforms

transform = transforms.Compose(
    [transforms.ToTensor(), transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4, shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

import matplotlib.pyplot as plt
import numpy as np
```

```
def imshow(img):
    img = img / 2 + 0.5    # unnormalize
    npimg = img.numpy( )
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
```

```
dataiter = iter(trainloader)
images, labels = dataiter.next()
```

```
imshow(torchvision.utils.make_grid(images))
print(' '.join('%5s' % classes[labels[j]] for j in range(4)))
```



```
import torch.nn as nn
import torch.nn.functional as F
```

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

```
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

```
net = Net( )
```

```
import torch.optim as optim
```

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters( ), lr=0.001, momentum=0.9)
```

```
for epoch in range(10): # 에포크 수
```

```
    running_loss = 0.0
```

```
    for i, data in enumerate(trainloader, 0):
```

```
        inputs, labels = data # 학습 데이터
```

```
        optimizer.zero_grad( )
```

```
        outputs = net(inputs)
```

```
        loss = criterion(outputs, labels)
```

```
        loss.backward( )
```

```
        optimizer.step( )
```

```
    running_loss += loss.item()
```

```
    if i % 2000 == 1999: # 매 2000 mini-batch 별로 출력
```

```
        print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 2000))
```

```
        running_loss = 0.0
```

```
print('Finished Training')
```

```
[9, 4000] loss: 0.723
[9, 6000] loss: 0.739
[9, 8000] loss: 0.783
[9, 10000] loss: 0.794
[9, 12000] loss: 0.799
[10, 2000] loss: 0.673
[10, 4000] loss: 0.708
[10, 6000] loss: 0.742
[10, 8000] loss: 0.748
[10, 10000] loss: 0.765
[10, 12000] loss: 0.772
Finished Training
```

```
dataiter = iter(testloader)
images, labels = dataiter.next( )
```

```
imshow(torchvision.utils.make_grid(images))
print('GroundTruth: ', ' '.join('%5s' % classes[labels[j]] for j in range(4)))
```



```
outputs = net(images)
_, predicted = torch.max(outputs, 1)
print('Predicted: ', ' '.join('%5s' % classes[predicted[j]] for j in range(4)))
```

```
Predicted:   cat  ship truck plane
```

```
correct = 0
total = 0
with torch.no_grad( ):
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
```

```
print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / total))
```

```
Accuracy of the network on the 10000 test images: 62 %
```



```
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))
```

```
with torch.no_grad( ):
```

```
    for data in testloader:
```

```
        images, labels = data
```

```
        outputs = net(images)
```

```
        _, predicted = torch.max(outputs, 1)
```

```
        c = (predicted == labels).squeeze()
```

```
        for i in range(4):
```

```
            label = labels[i]
```

```
            class_correct[label] += c[i].item()
```

```
            class_total[label] += 1
```

```
for i in range(10):
```

```
    print('Accuracy of %5s : %2d %%' % (classes[i], 100 * class_correct[i] / class_total[i]))
```

```
Accuracy of plane : 69 %
Accuracy of   car : 80 %
Accuracy of  bird : 52 %
Accuracy of   cat : 36 %
Accuracy of  deer : 53 %
Accuracy of   dog : 52 %
Accuracy of  frog : 68 %
Accuracy of horse : 67 %
Accuracy of  ship : 75 %
Accuracy of truck : 70 %
```

Quiz

1. 1x1 컨볼루션에서 컨볼루션 필터는 깊이방향의 1차원 배열 형태를 갖는다. (O,X)
2. 컨볼루션 층의 출력으로 만들어지는 채널 개수는 컨볼루션 필터의 개수와 같다. (O,X)
3. 디컨볼루션이 적용되면 입력보다 더 큰 출력이 만들어질 수 있다. (O,X)
4. 팽창 컨볼루션을 적용하면 더 넓은 영역의 특징을 추출할 수도 있다. (O,X)
5. 공간 분할 컨볼루션인 경우 학습되어야 하는 컨볼루션 필터의 파라미터가 더 많다. (O,X)
6. 깊이별 분할 컨볼루션은 채널별로 컨볼루션을 한 다음, 1x1 컨볼루션을 적용한다. (O,X)