컨볼루션 연산과 영상 분류 CNN 모델

이건명
충북대학교 산업인공지능학과

학습 내용

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

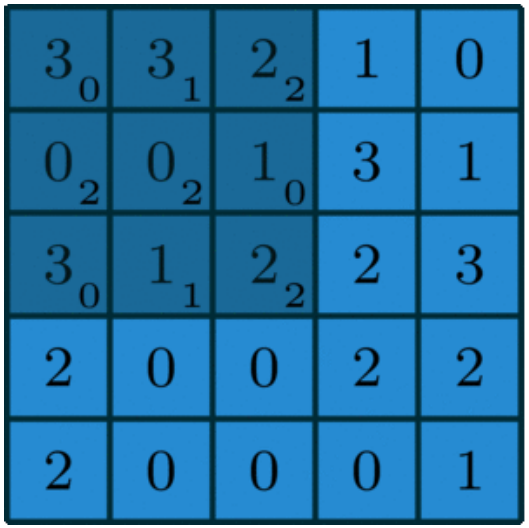
1. 컨볼루션의 형태

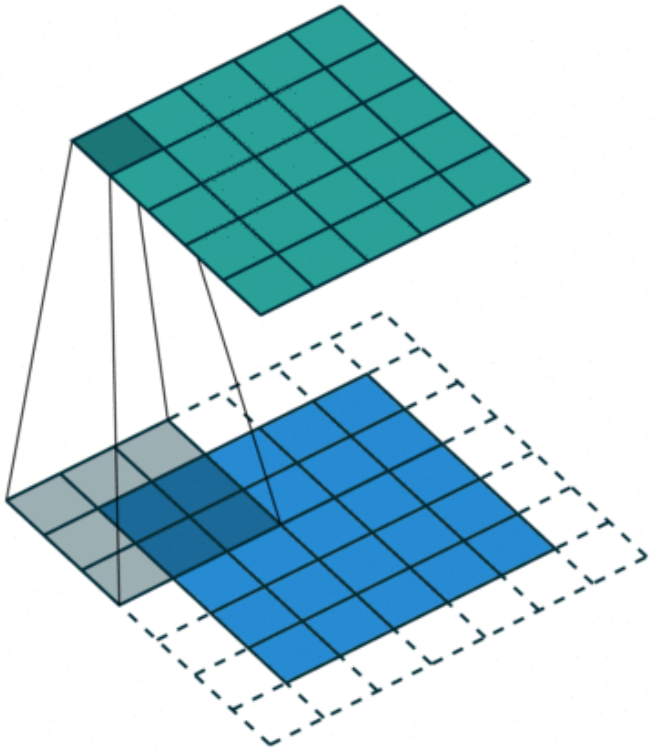
- 단일 채널 컨볼루션
- 다중 채널 2D 컨볼루션
- 다중 채널 3D 컨볼루션
- 1x1 컨볼루션
- 디컨볼루션
- 팽창 컨볼루션
- 공간 분할 컨볼루션
- 깊이별 분할 컨볼루션
- 집단 컨볼루션
- 채널섞기 집단 컨볼루션

컨볼루션

❖ 단일 채널 대상의 컨볼루션

- 컨볼루션 커널, 스트라이드(stride), 패딩(padding)

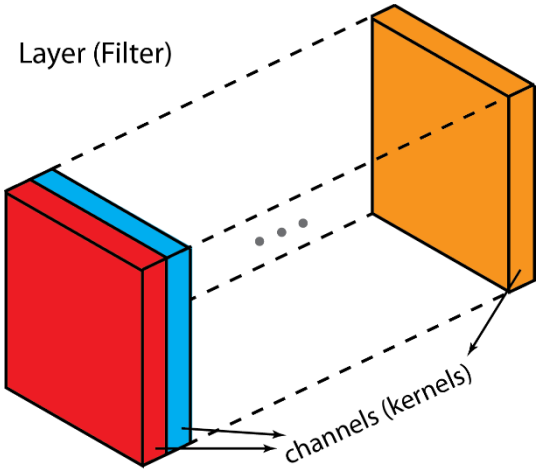


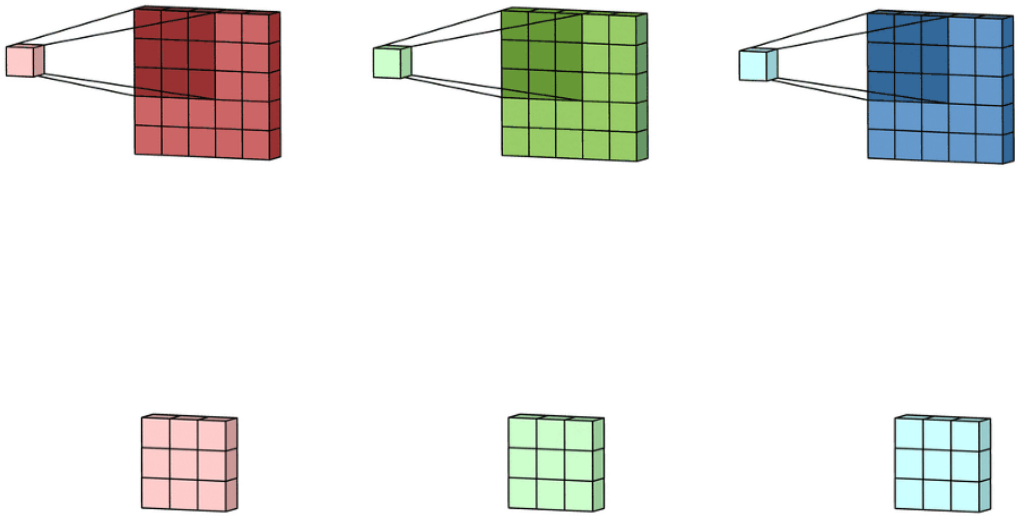


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 다중 채널의 2D 컨볼루션

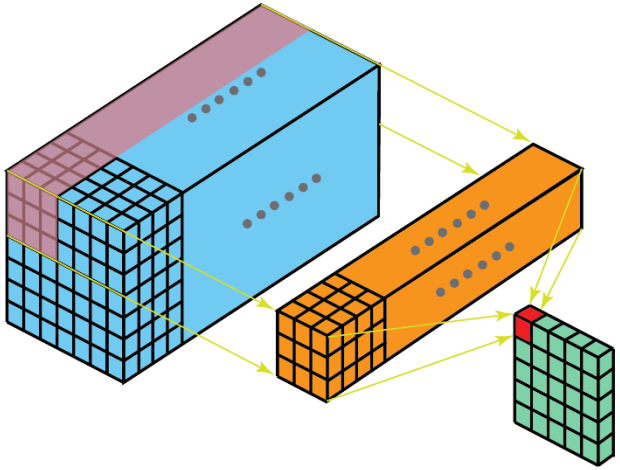


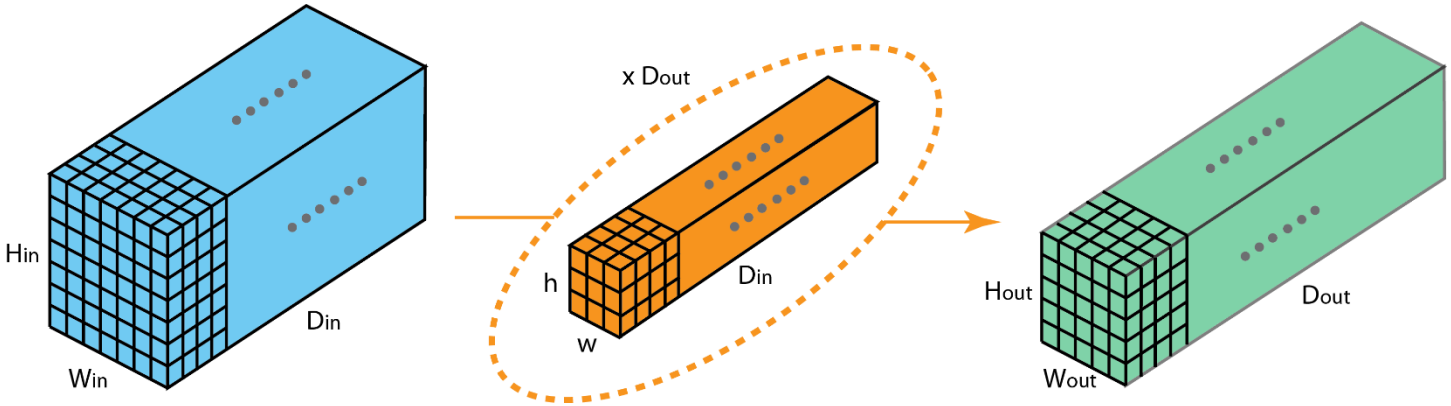


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 다중 채널의 2D 컨볼루션 – cont.

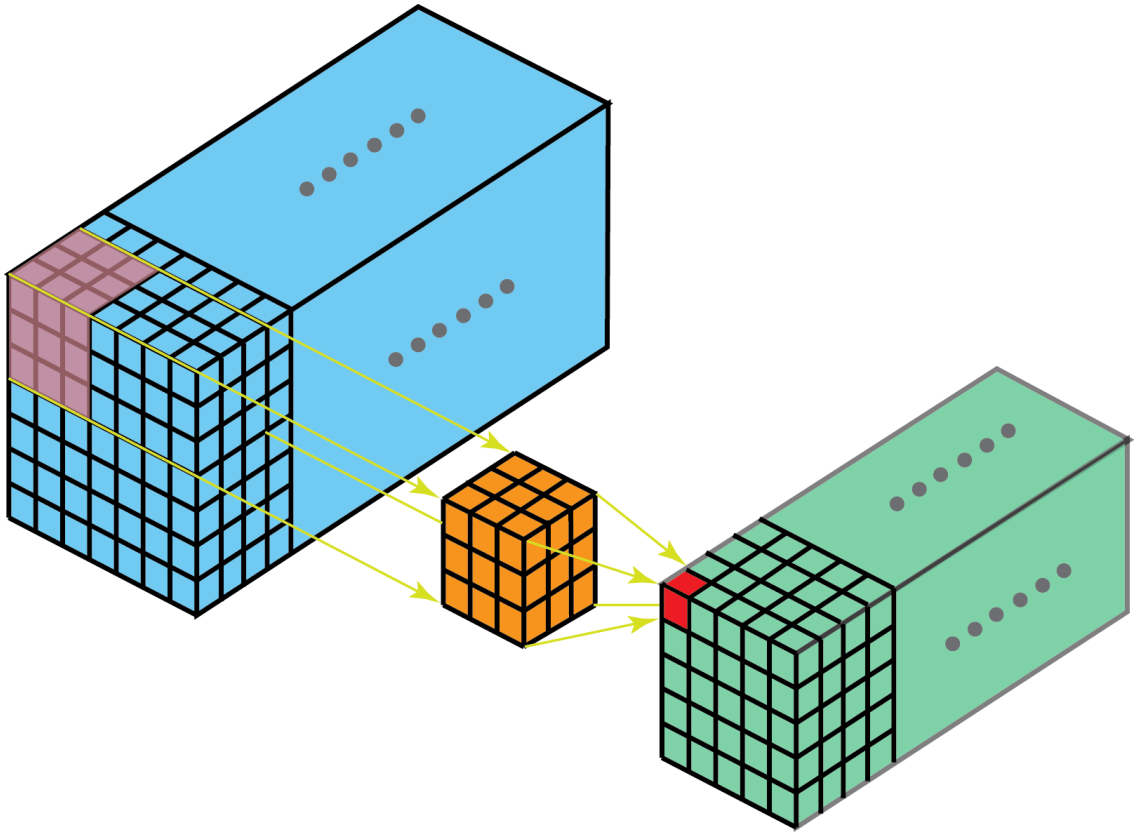




https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

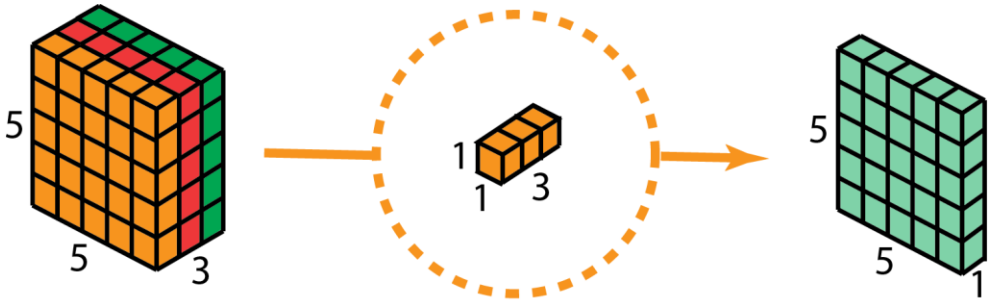
❖ 3D 컨볼루션

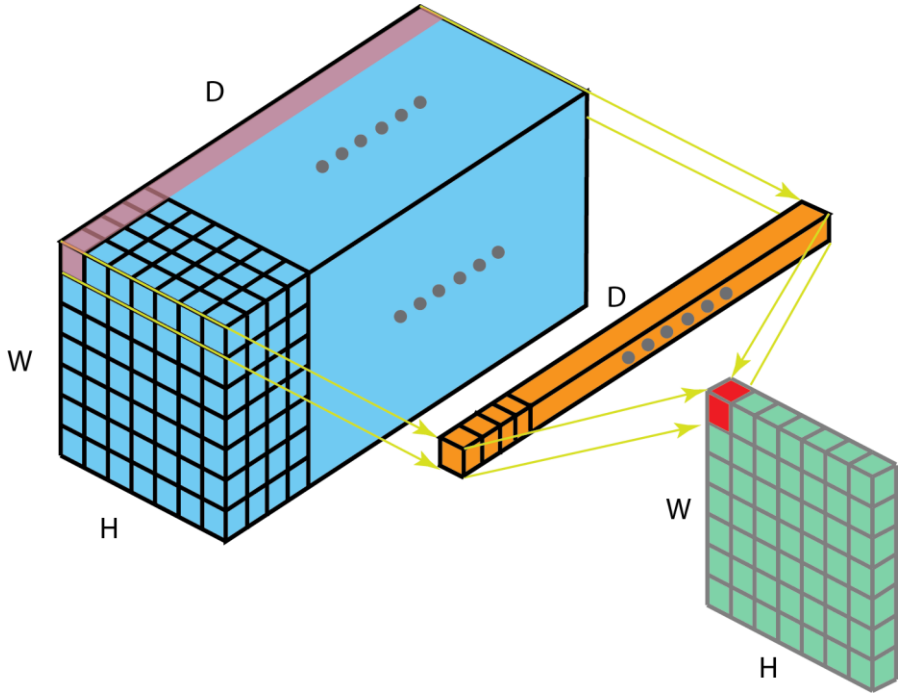


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 1 x 1 컨볼루션



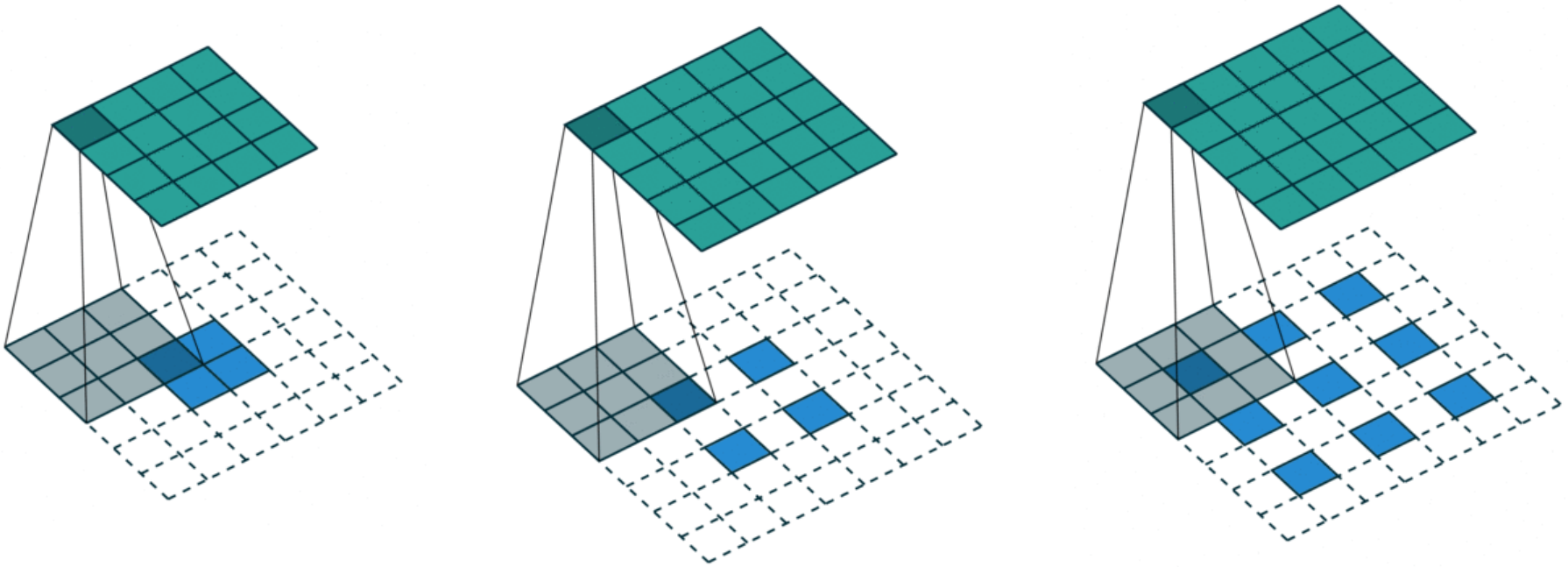


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

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

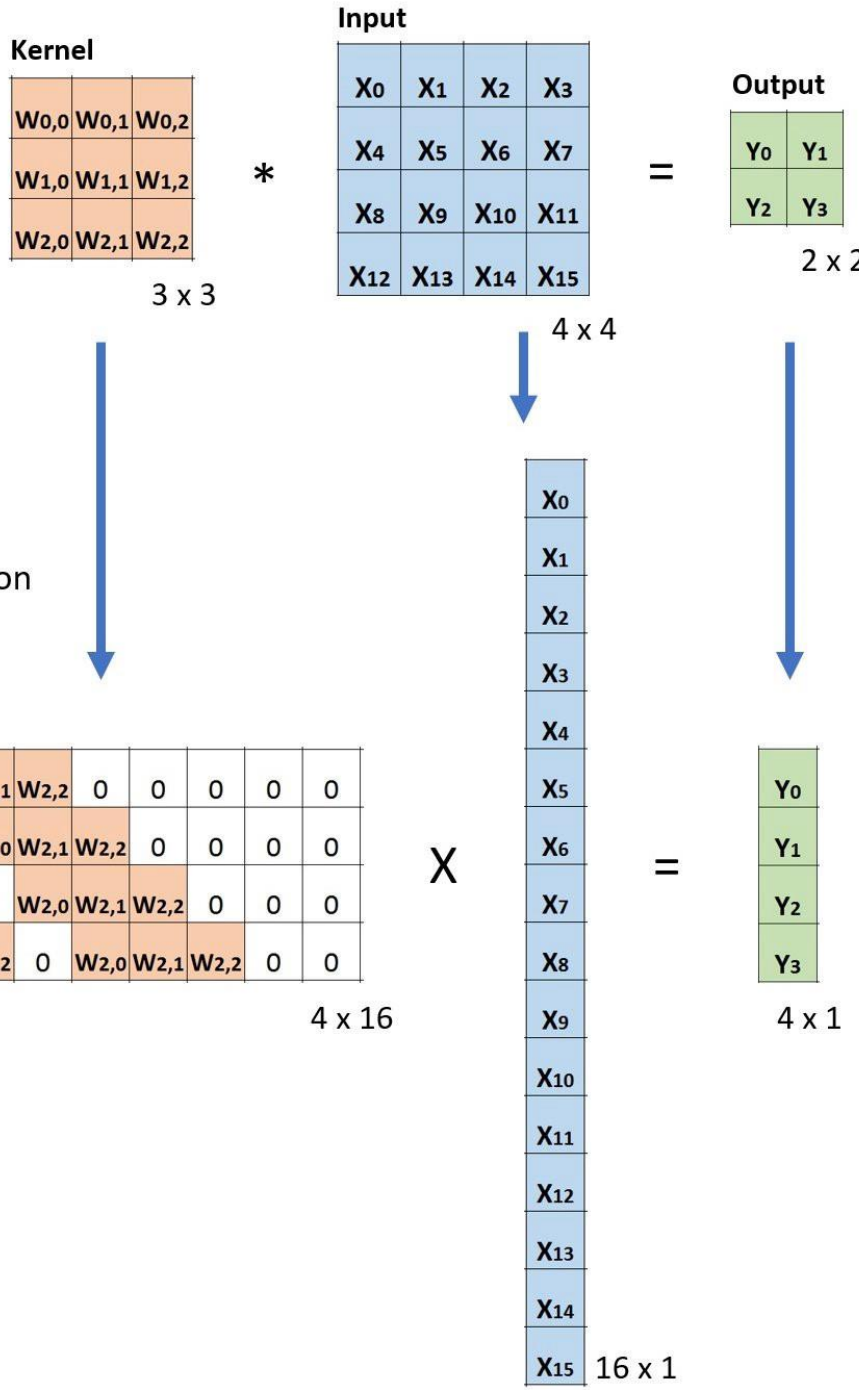
- 입력보다 큰 출력 생성(upsampling)



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

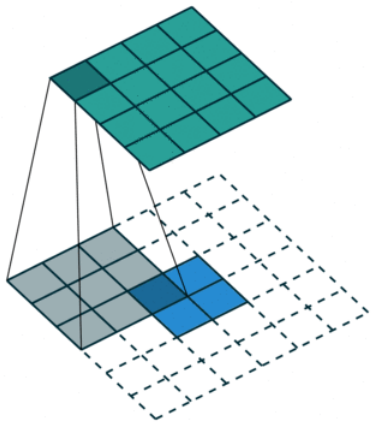
❖ 일반적인 컨볼루션



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 디컨볼루션



transposed convolution 이름의 유래

$W_{0,0}$	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

X

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

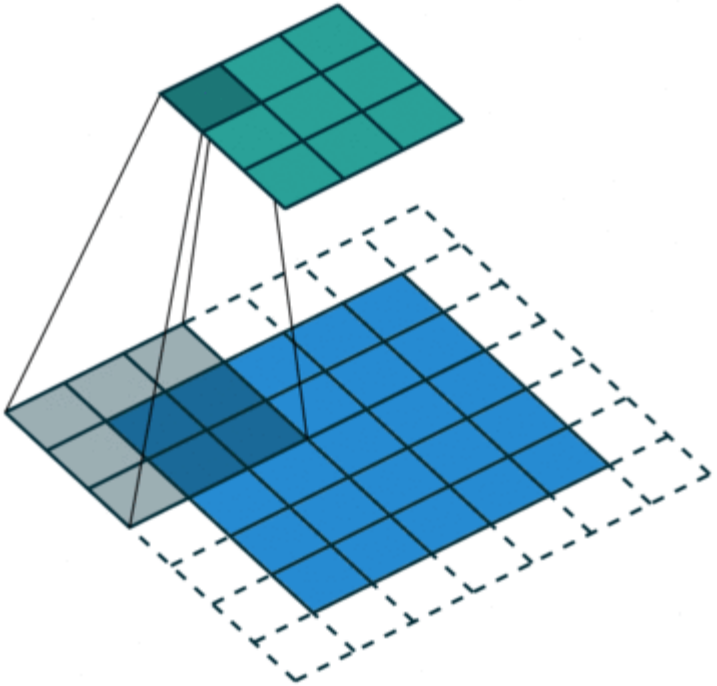
X_0	X_1	X_2	X_3
X_4	X_5	X_6	X_7
X_8	X_9	X_{10}	X_{11}
X_{12}	X_{13}	X_{14}	X_{15}

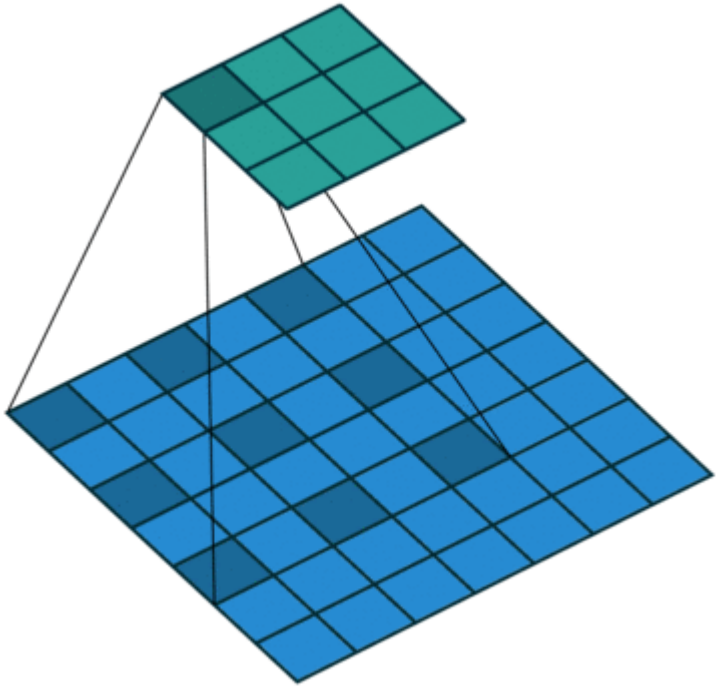
4 x 4

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 팽창 컨볼루션(dilated convolution, atrous convolution)



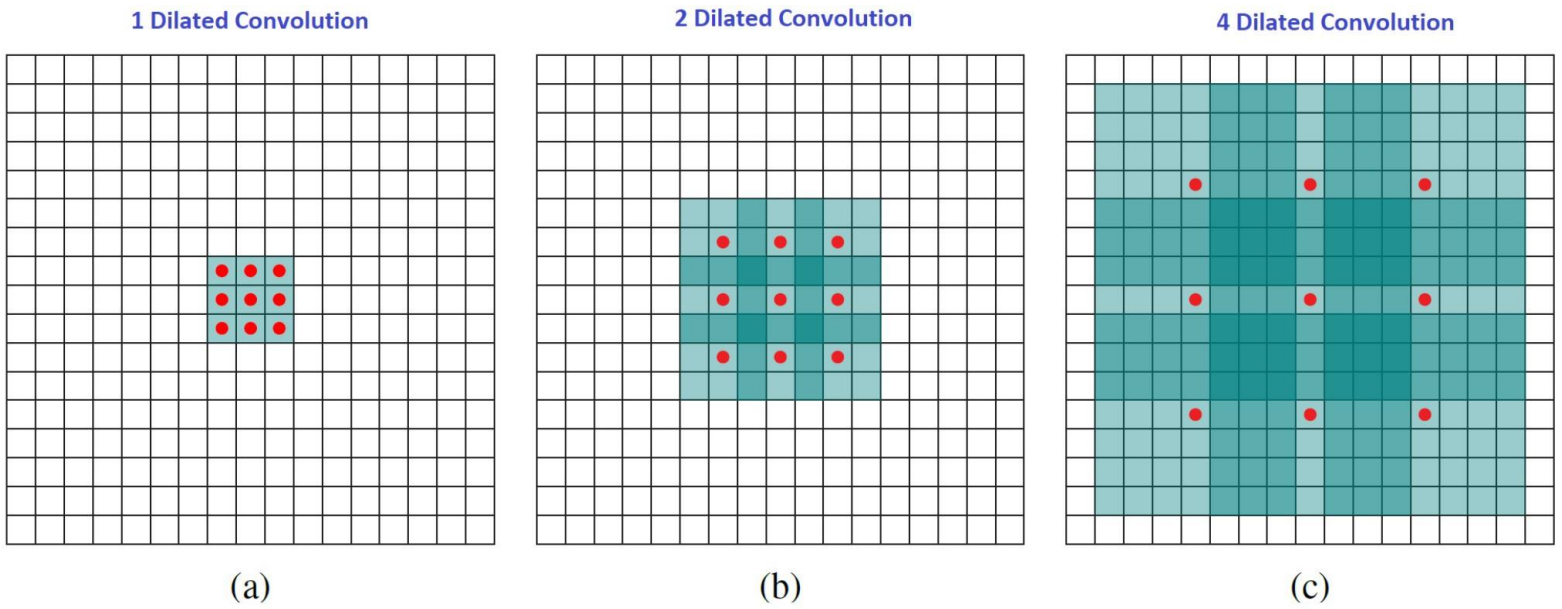


a trous (with holes)

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

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



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 분할(separable) 컨볼루션

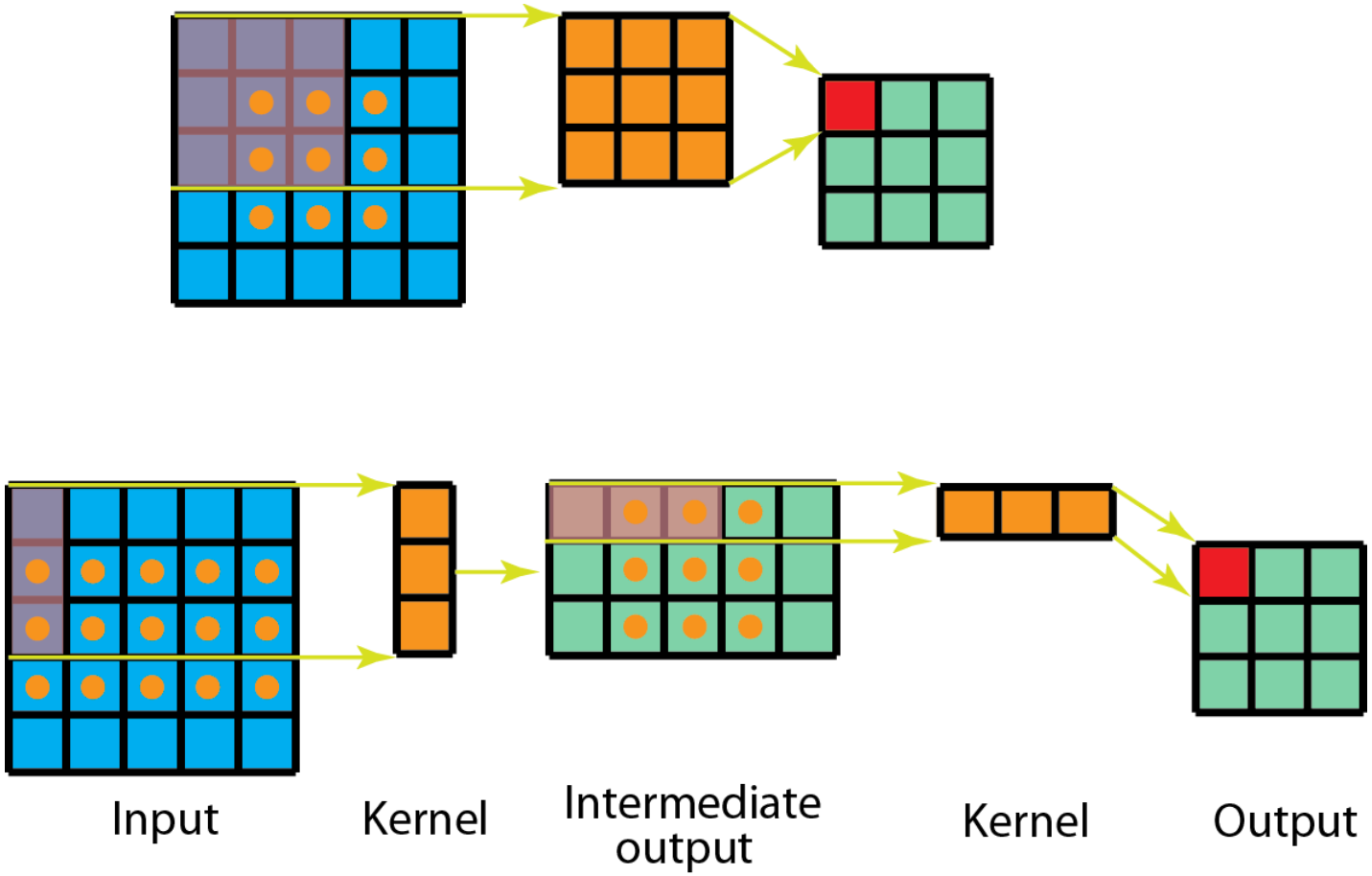
- 공간 분할 컨볼루션
- 깊이별 분할 컨볼루션

❖ 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)

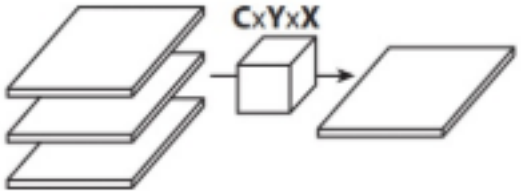


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

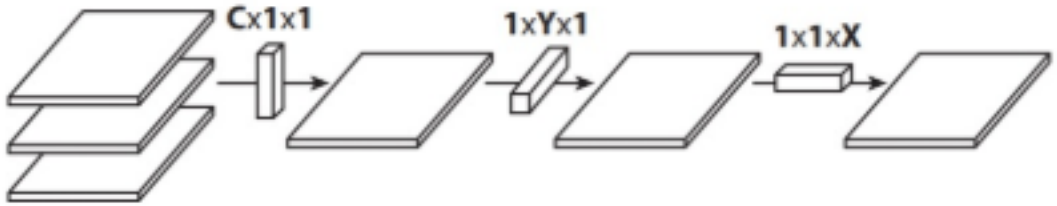
컨볼루션

❖ 공간 분할 컨볼루션

- **평탄화 컨볼루션(flattened convolution)**



3D 컨볼루션

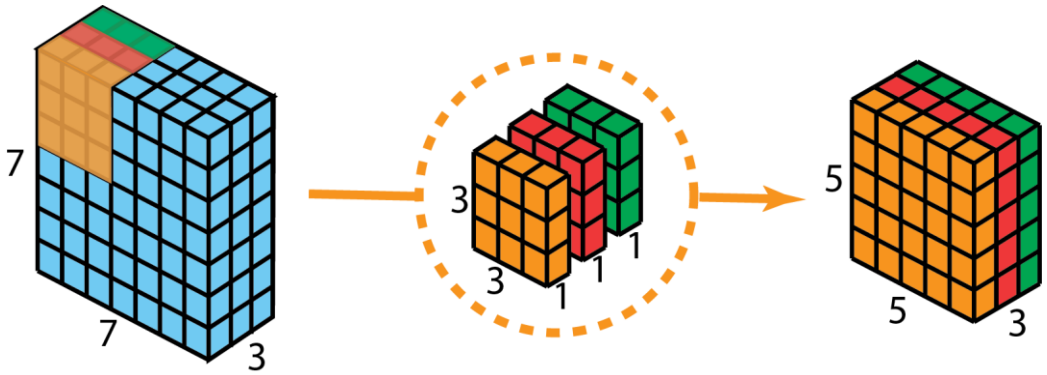


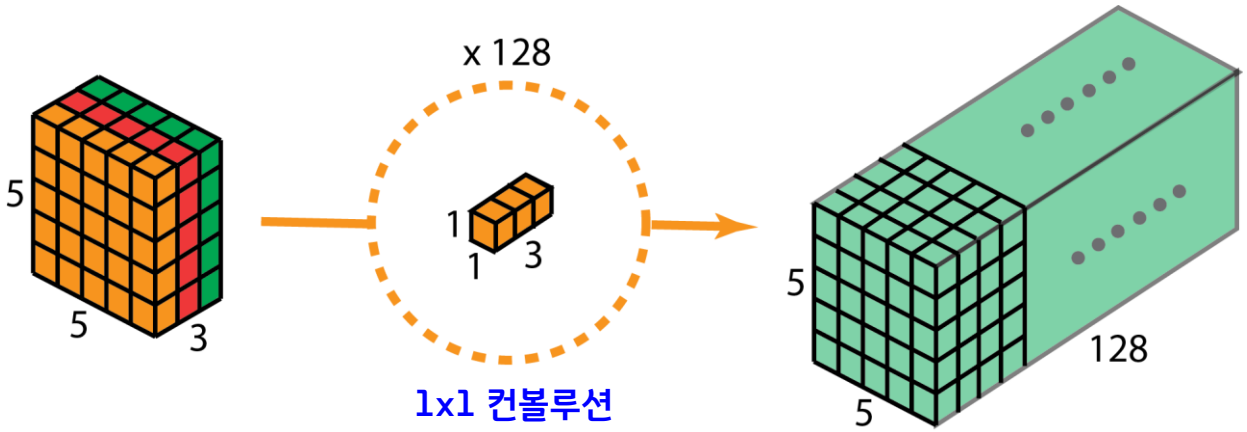
다른 방향의 1D 컨볼루션

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 깊이별 분할 컨볼루션(Depthwise separable convolution)



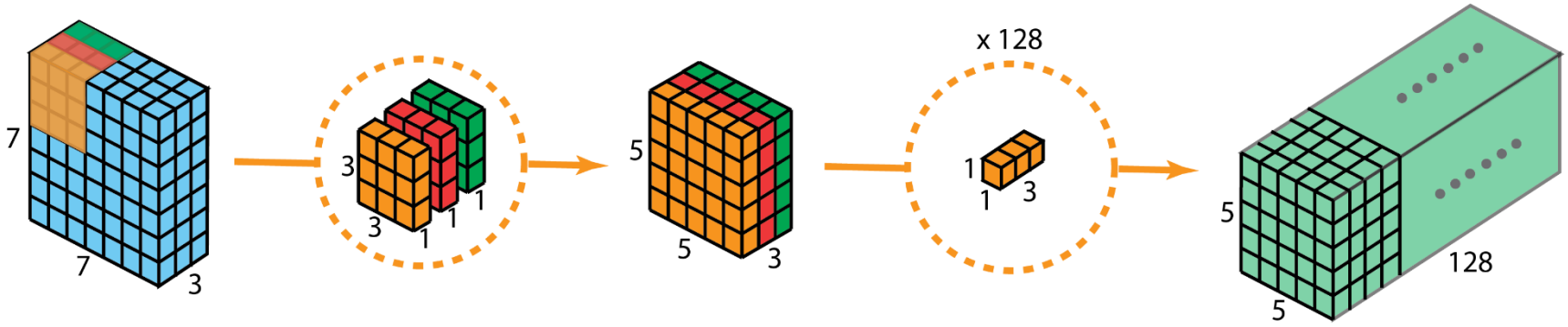


1x1 컨볼루션

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

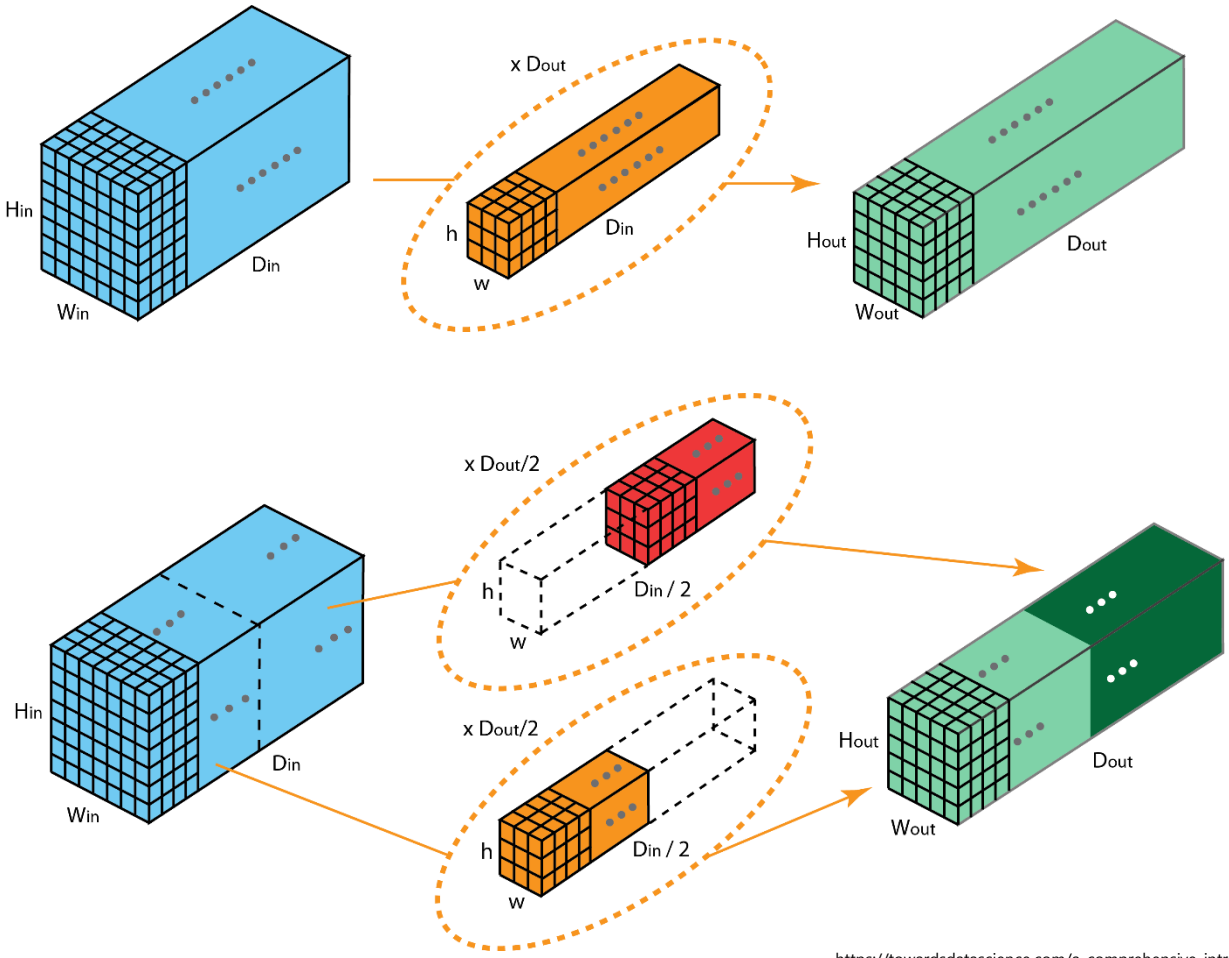
❖ 깊이별 분할 컨볼루션(Depthwise separable convolution) – cont.

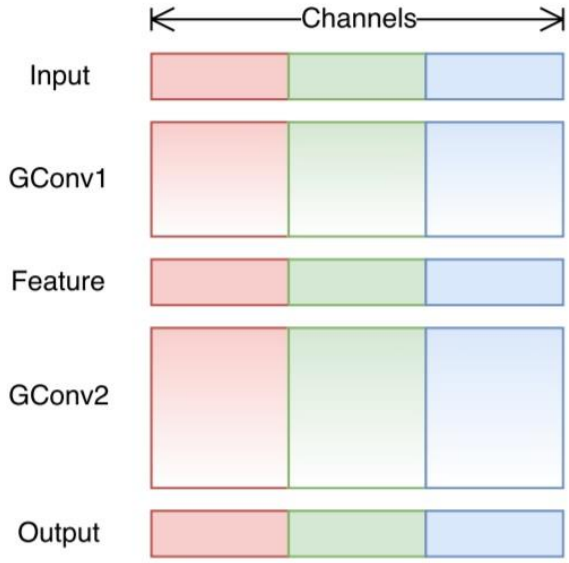


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

❖ 집단 컨볼루션(grouped convolution)

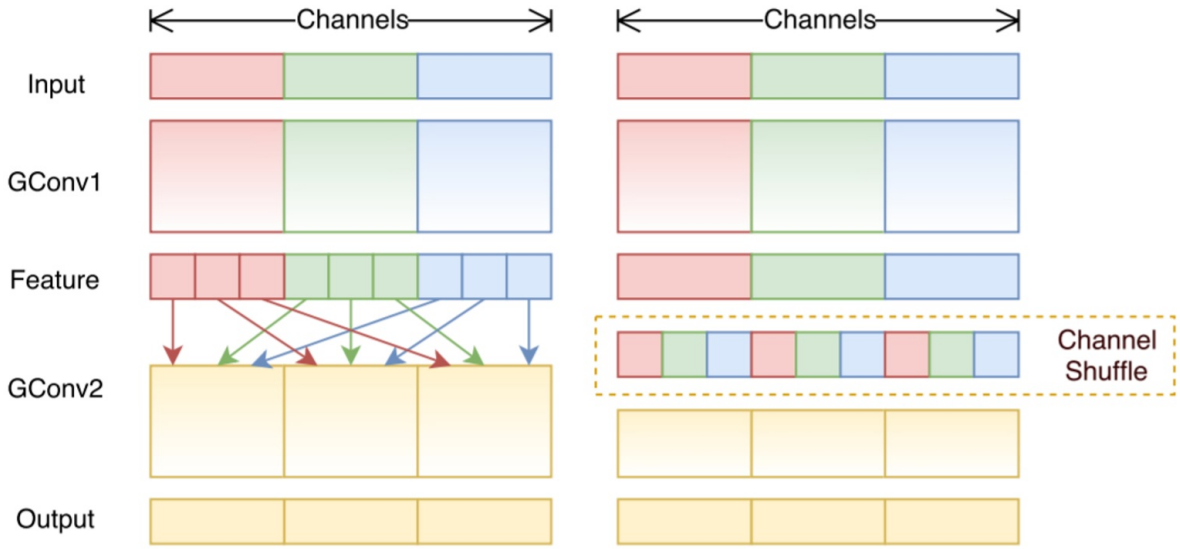




https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

컨볼루션

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



https://ikhlestov.github.io/pages/machine-learning/convolutions-types/

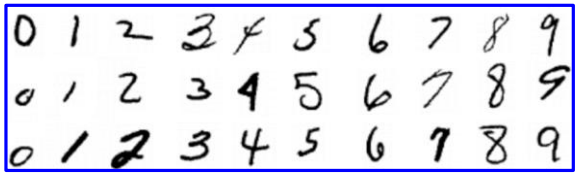
2. 물체 인식 CNN 모델

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

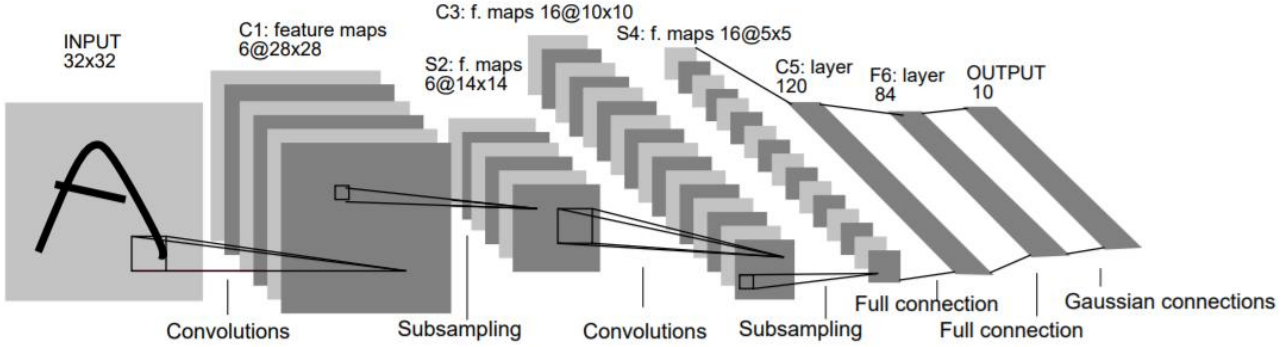
LeNet 모델

❖ 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%)

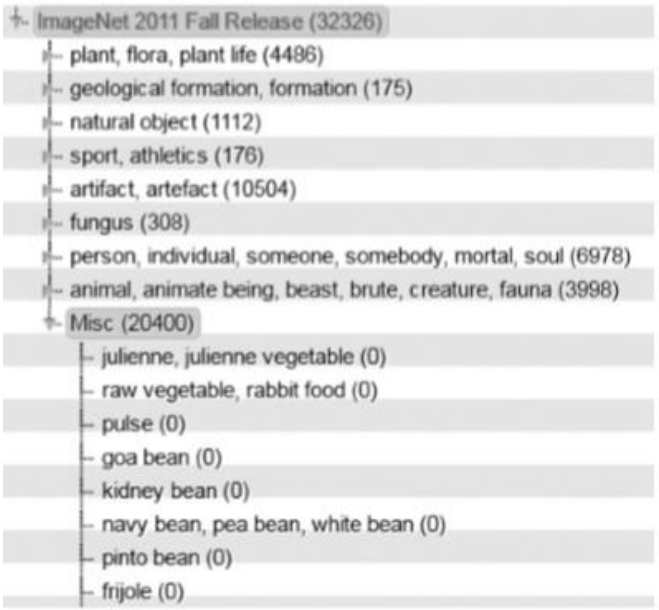


http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

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

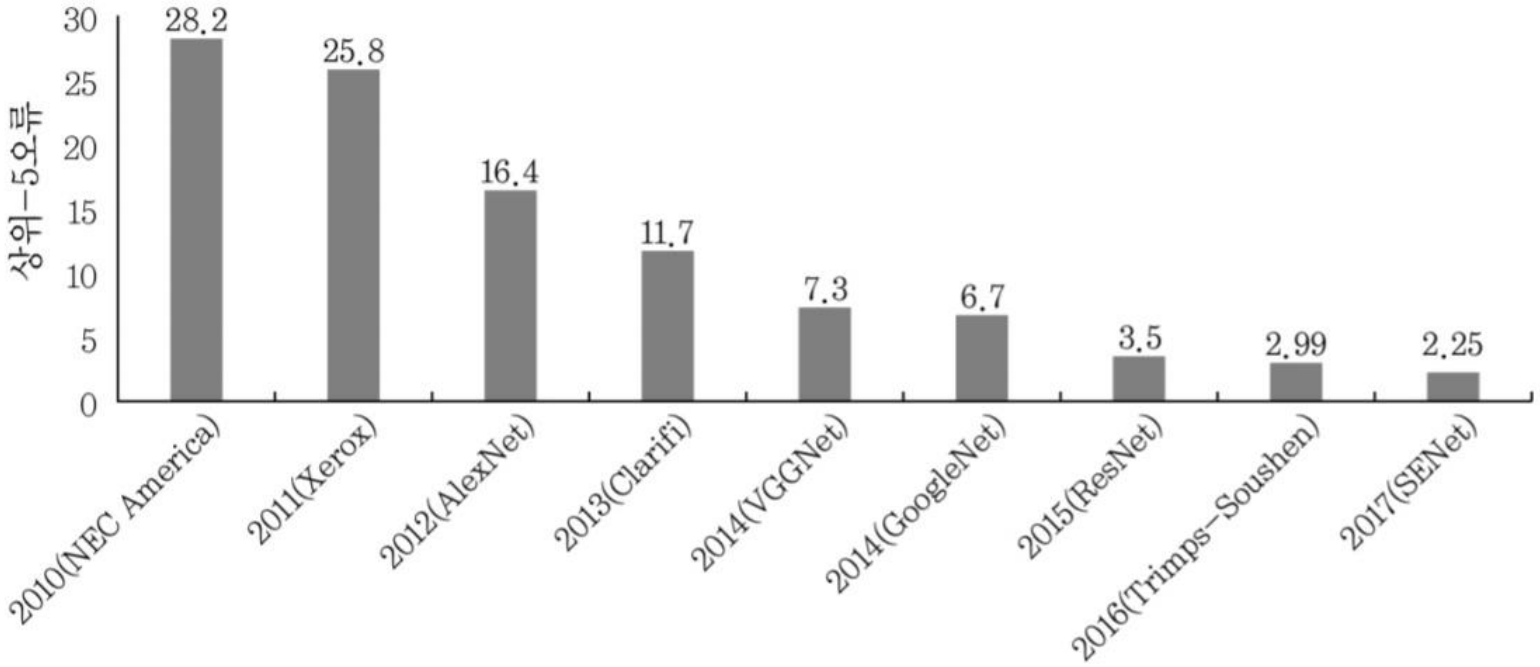


그림 5.19 ILSVRC 주요 우수팀의 성적

가로축은 연도, 괄호 안에는 팀 이름이나 모델 이름을 나타냄

AlexNet 모델

❖ AlexNet

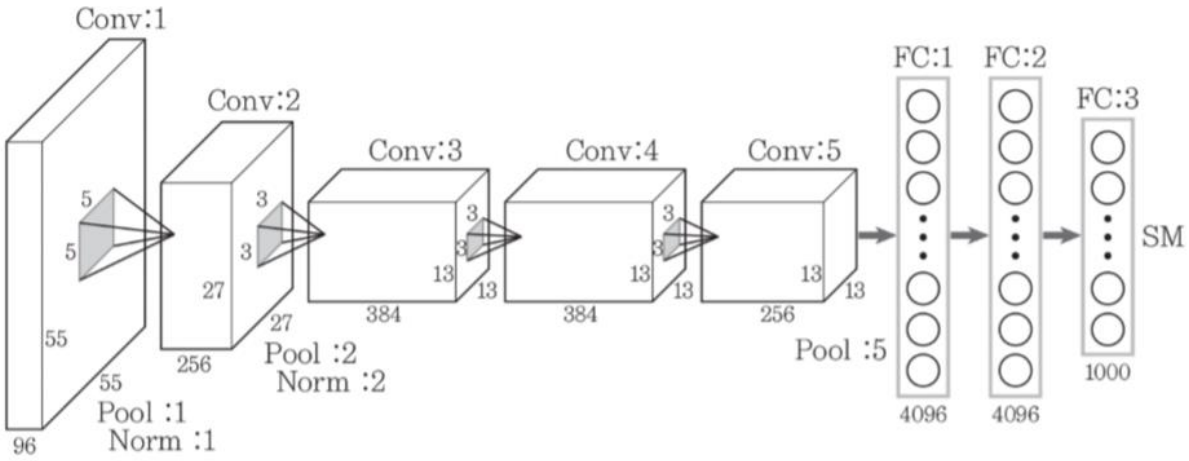
- 토론토 대학 Geoffrey E. Hinton 팀이 제안
- ILSVRC에서 2012년 우승
- **상위-5 오류율 : 16.43%**
  - 직전 년도 대비 9.4% 정확도 향상

http://www.image-net.org/

AlexNet 모델

❖ **AlexNet – cont.**

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

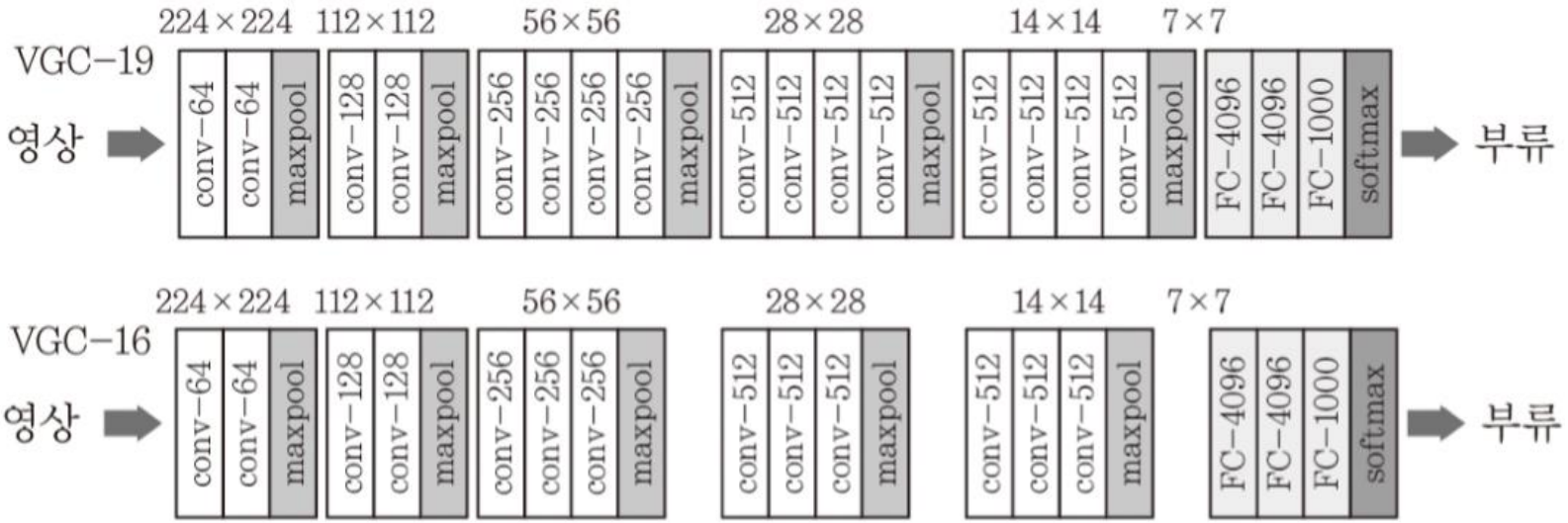


- **ReLU 함수**를 사용한 첫 모델
- FC 층에 **드롭아웃**(dropout) 기법 사용
- **최대값 풀링**(max pooling) 사용

VGGNet 모델

❖ VGGNet

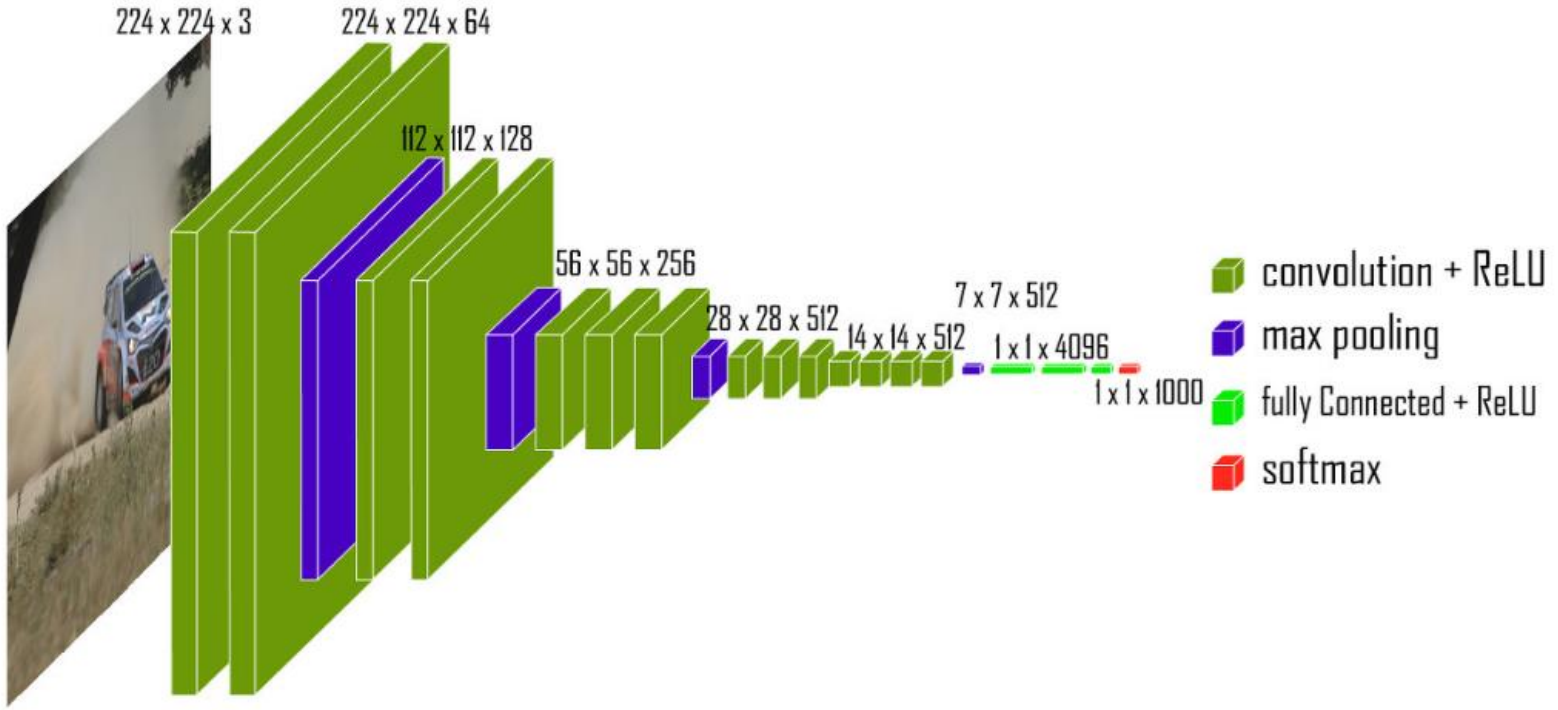
- VGG-16 모델(16개 층)
- VGG-19 모델(19개 층)
- 2014년 ILSVRC에서 2등 차지 (상위-5 오류율: 7.32%)
- 단순한 구조 (3x3 커널 사용)



VGGNet 모델

❖ VGGNet – cont.

- VGG-16 구조의 계층별 특징지도



https://www.geeksforgeeks.org/vgg-16-cnn-model/

VGGNet 모델

❖ VGGNet – cont.

- 모든 층에서 3x3 필터 사용
- 3x3 필터 2회 적용 ⇒ 5x5 필터 적용 효과
- 3x3 필터 3회 적용 ⇒ 7x7 필터 적용 효과

27 가중치 49 가중치

ReLU 3회 적용 ⇒ 복잡한 결정경계 표현 가능

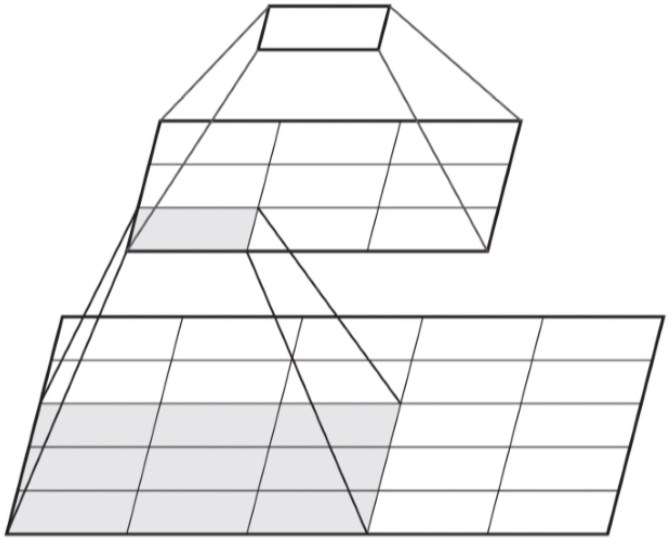
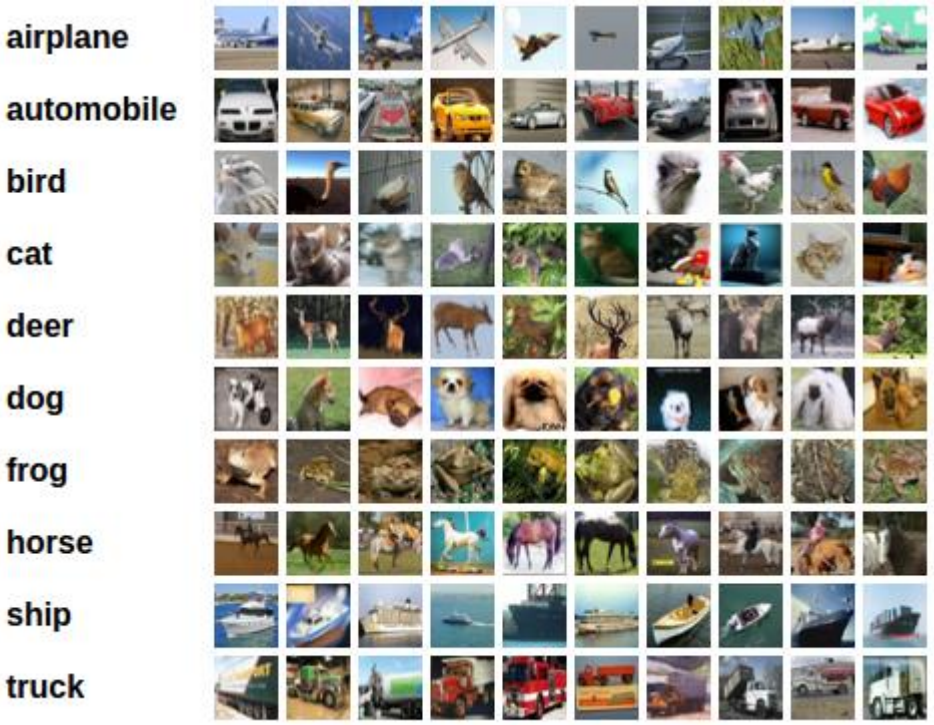


그림 5.23 2개 층의 3×3 컨볼루션에 의한 5×5 컨볼루션 구현

[실습] CIFAR10 데이터의 인식

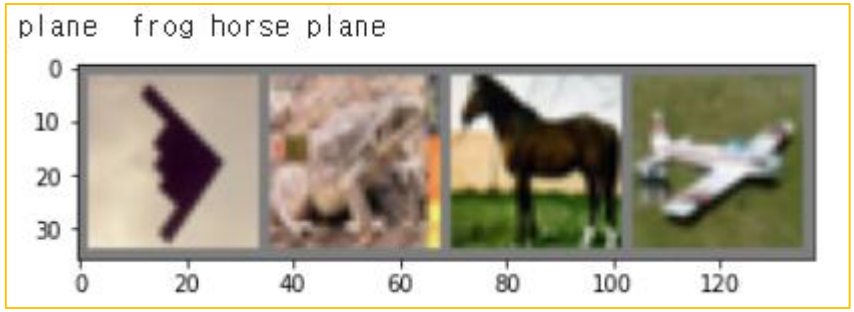
❖ CIFAR 10 데이터

- 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)))
```



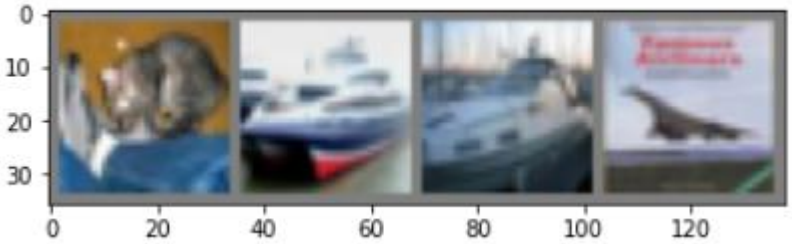
```
import torch.nn as nn
import torch.nn.functional as F

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)))
```

```
GroundTruth:    cat  ship  ship plane
```




```
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 %
```

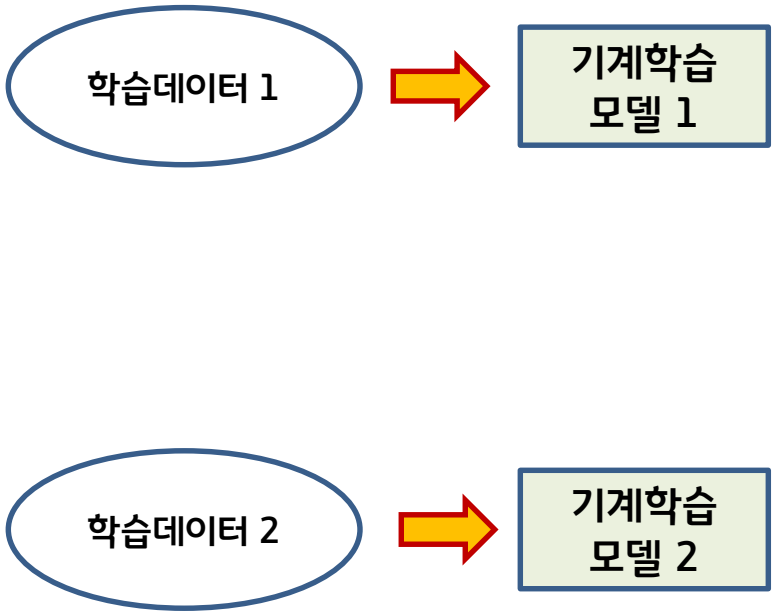
3. 전이 학습

❖ 전이 학습(transfer learning)

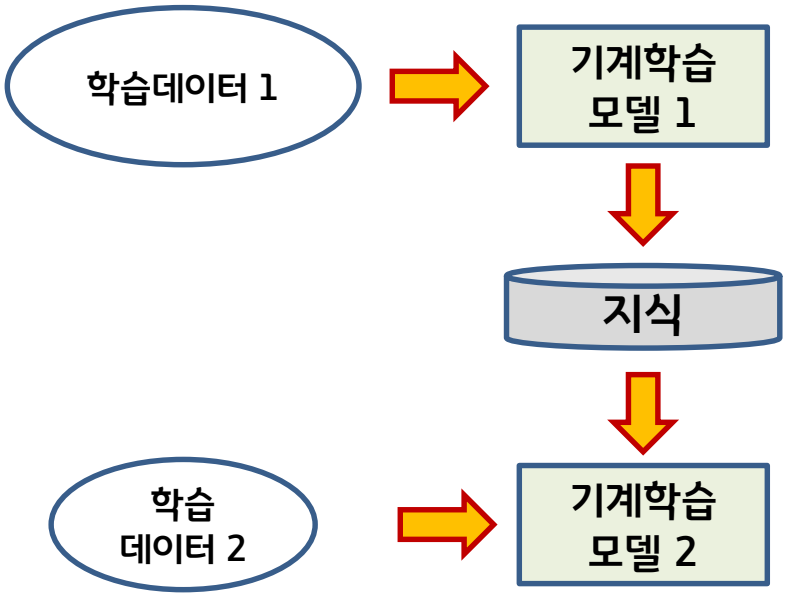
- 큰 규모의 딥러닝 신경망을 학습시킬 때는, 많은 학습 데이터와 상당한 학습 시간이 필요
- 대규모 영상 데이터베이스인 ImageNet 데이터를 학습한 여러 컨볼루션 신경망 모델 공개
- 공개된 모델을 가져다가 누구나 자신의 문제가 적용해 볼 수도 있고, 모델의 일부 활용 가능
- 학습된 컨볼루션 신경망의 컨볼루션 층들을 가져오고 뒤 단계에서 분류하는 다층 퍼셉트론 모델을 붙여서 학습

전이 학습

❖ 전통적 기계학습

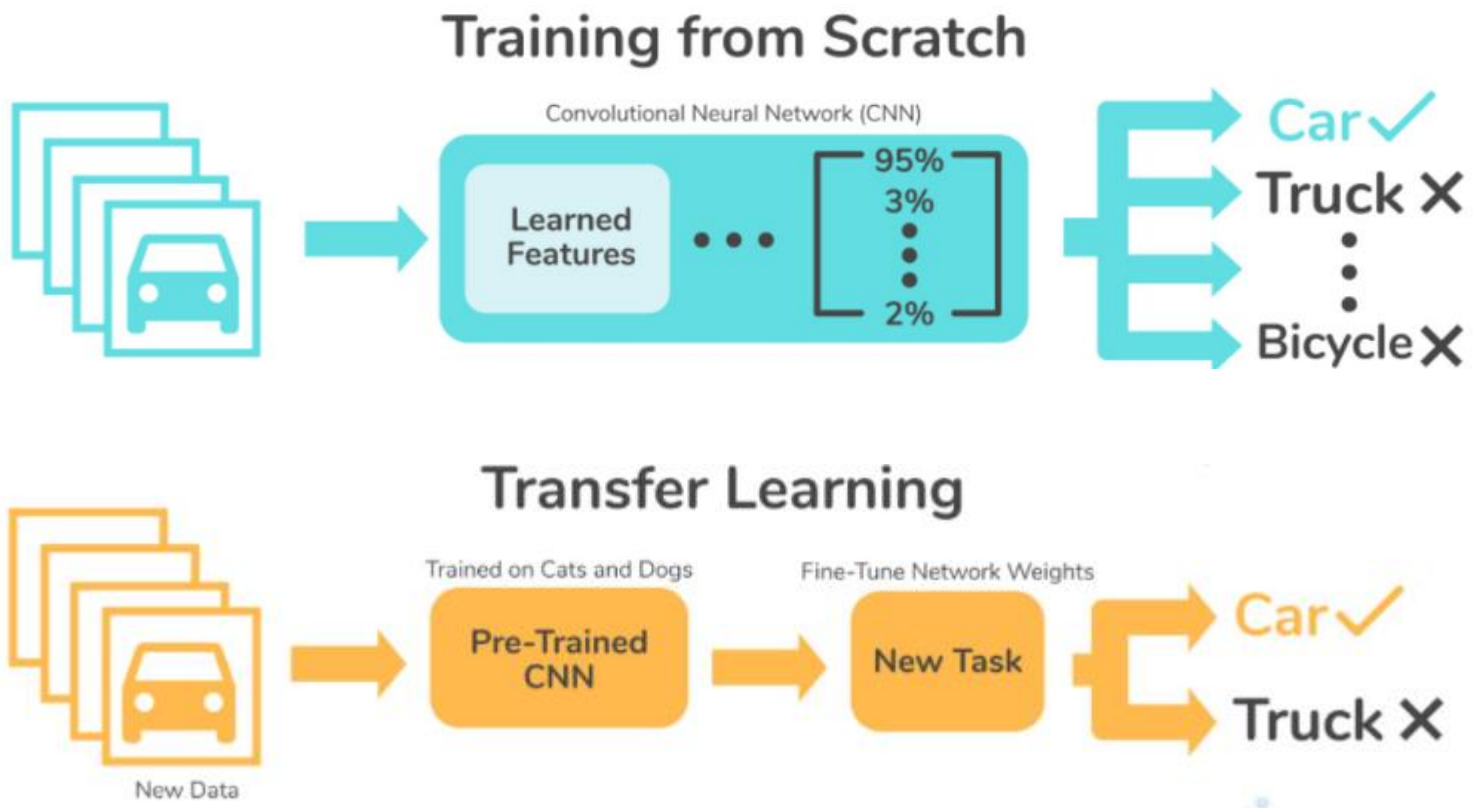


전이 학습



전이 학습

❖ 딥러닝 모델의 전이 학습

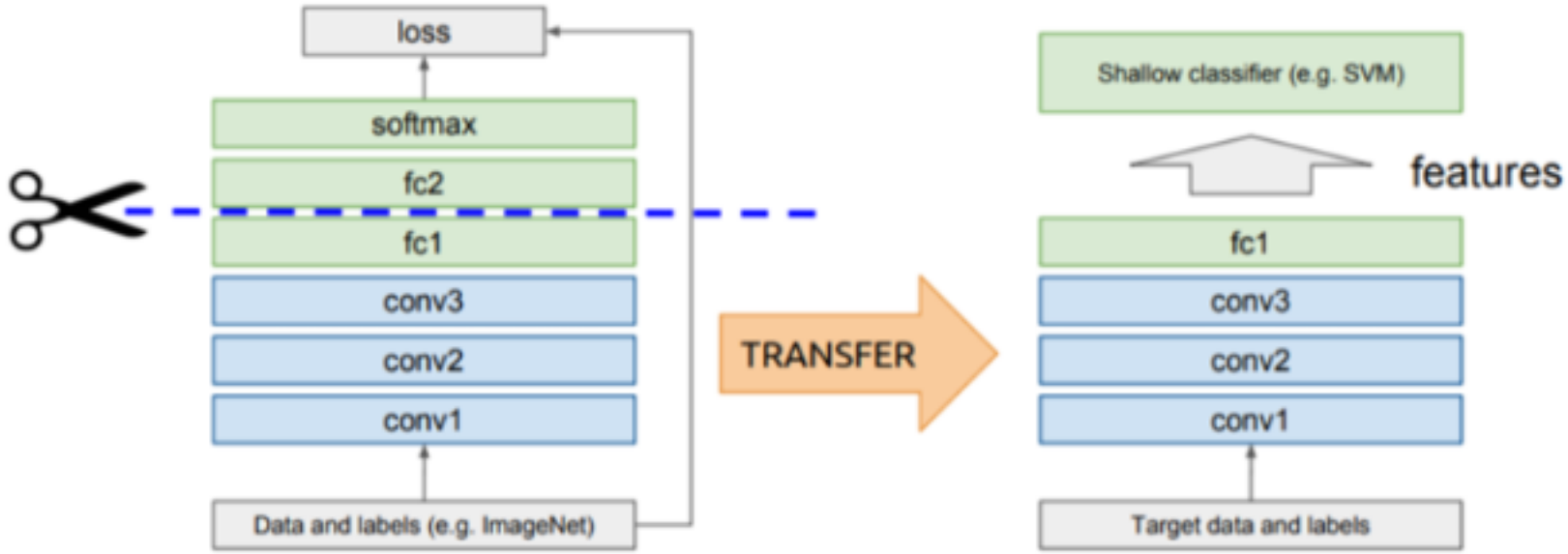


https://www.freetimelearning.com/software-interview-questions-and-answers.php?Explain-transfer-learning-in-the-context-of-deep-learning.&id=4184

전이 학습

❖ 딥러닝 모델의 전이 학습

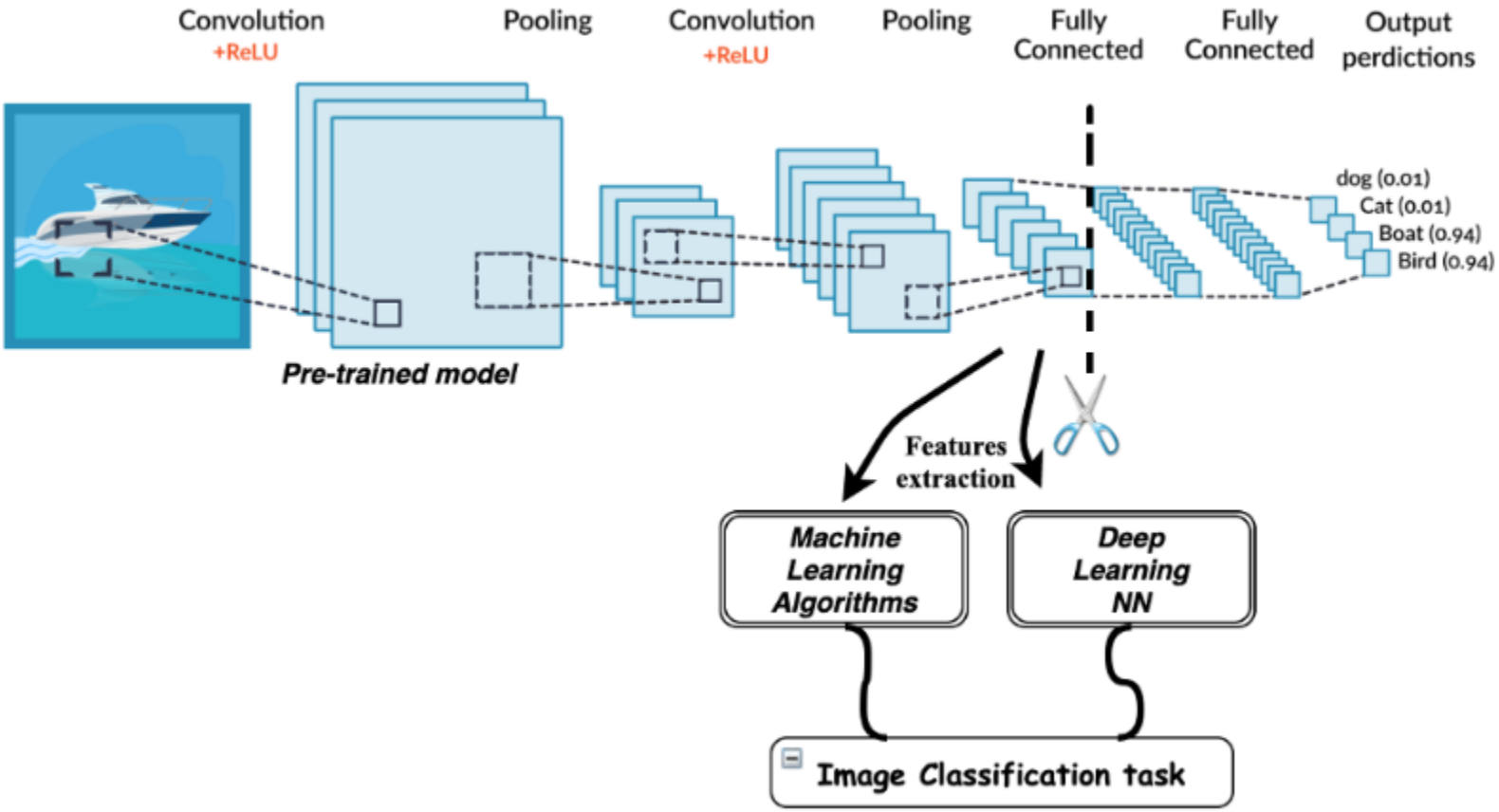
- 예. 사전 학습 모델의 특징 추출기 활용



https://skyengine.ai/se/skyengine-blog/128-what-is-transfer-learning

전이 학습

❖ 전이 학습의 형태



https://medium.com/tag/deep-learning/latest

전이 학습

❖ PyTorch 모델

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet = models.mobilenet_v2(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

Quiz

❖ **다중 채널 2D 컨볼루션의 특징은?**

① 단일 채널보다 계산량이 적다
② 여러 채널을 동시에 처리
③ 이미지의 색상을 변경
④ 이미지를 회전

❖ **1x1 컨볼루션의 주요 목적은?**

① 이미지의 크기 조정
② 채널 간의 상호작용 학습
③ 이미지의 회전
④ 이미지 압축

❖ **디컨볼루션의 주된 용도는?**

① 이미지의 크기를 줄임
② 이미지를 흐리게 함
③ 이미지의 크기를 늘림
④ 이미지를 회전

Quiz

❖ 팽창 컨볼루션의 장점은?

① 더 작은 영역의 특징을 추출

② 더 넓은 영역의 특징을 추출

③ 이미지를 회전

④ 이미지를 압축

❖ 깊이별 분할 컨볼루션의 단계는?

① 3x3 컨볼루션 후 5x5 컨볼루션 적용

② 채널별 컨볼루션 후 1x1 컨볼루션 적용

③ 1x1 컨볼루션 후 3x3 컨볼루션 적용

④ 5x5 컨볼루션 후 1x1 컨볼루션 적용

❖ LeNet 모델의 주요 특징은?

① 10 계층 구조

② 3x3 컨볼루션 사용

③ 5 계층 구조

④ ReLU 활성화 함수 사용

Quiz

❖ AlexNet 모델이 처음 사용한 활성화 함수는?

① Sigmoid
② ReLU
③ Tanh
④ Softmax

❖ VGGNet 모델의 주요 구조적 특징은?

① 1x1 컨볼루션 사용
② 3x3 필터 사용
③ 5x5 필터 사용
④ 7x7 필터 사용

❖ ILSVRC 대회에서 평가되는 주요 기준은?

① 최상위-1 오류율
② 최상위-3 오류율
③ 최상위-5 오류율
④ 최상위-10 오류율

Quiz

❖ 전이학습(Transfer Learning)이란?

① 학습 데이터를 증강하는 기법

② 한 모델의 학습된 가중치를 다른 모델에 사용하는 기법

③ 모델의 학습 속도를 높이는 최적화 기법

④ 모델의 복잡도를 줄이는 정규화 기법

❖ 다음 중 전이학습을 활용할 때 가장 적합한 시나리오는?

① 데이터가 충분하고 새로운 데이터가 기존 데이터와 전혀 관련이 없을 때
② 데이터가 부족하고 새로운 데이터가 기존 데이터와 매우 유사할 때
③ 데이터 처리 없이 바로 사용하고 싶을 때
④ 모든 데이터를 처음부터 새로 학습시키고 싶을 때

❖ 다음 중 전이학습의 주요 이점이 아닌 것은?

- 학습 시간을 단축시킬 수 있다.
- 적은 양의 데이터로도 높은 성능을 낼 수 있다.
- 항상 모든 문제에 대해 최상의 성능을 보장한다.
- 자원이 제한된 환경에서 유용하게 사용될 수 있다.

Quiz

- 입력이 64개의 채널을 가지고 있고, 128개의 1x1 컨볼루션 필터를 사용하는 레이어의 출력 특성 맵의 깊이와 총 파라미터 수를 계산하시오.

- 10개의 입력 채널에 대해 각각 6개의 5x5 필터가 적용되는 컨볼루션 층의 총 가중치 수를 계산하시오.

- 입력 채널이 3개, 출력 채널이 64개인 33×3 컨볼루션 레이어를 32×32 크기의 이미지에 적용했을 때의 파라미터 수와 출력 크기를 계산하시오.