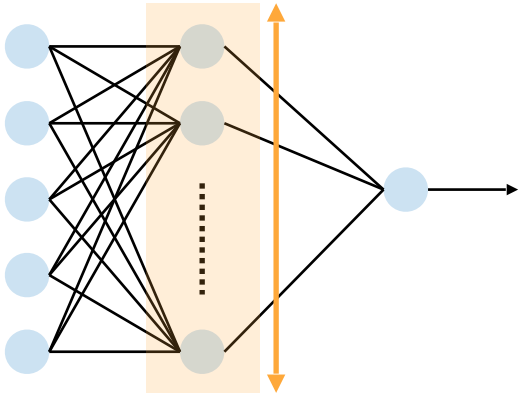
합성곱 신경망

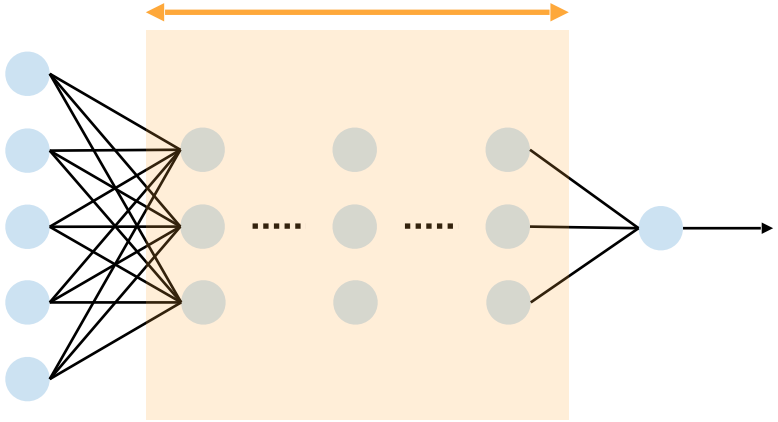
Convolutional Neural Network

정우일

5.1 합성곱 신경망의 발달 배경



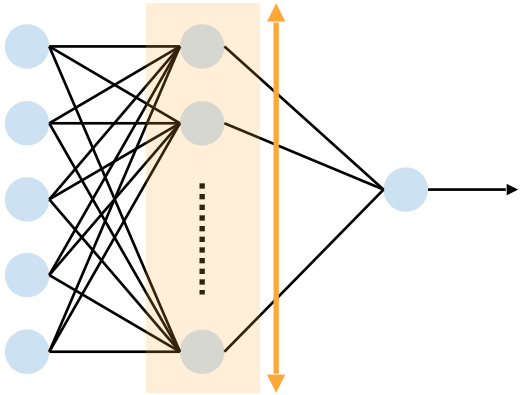
Single Hidden Layer Perceptron



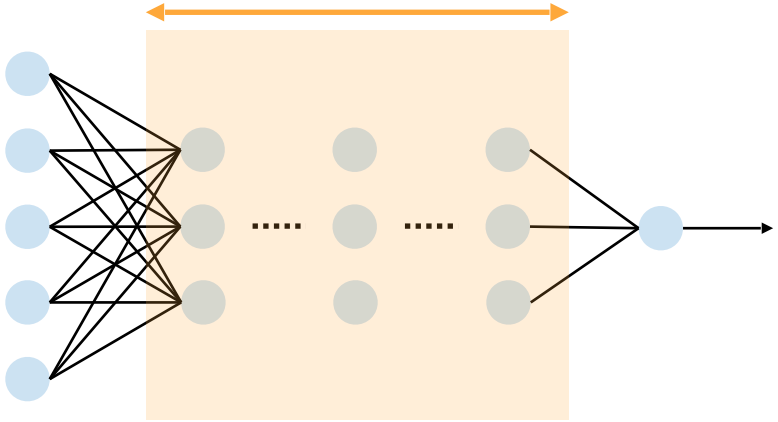
Multi Hidden Layer Perceptron

보편 근사 정리(Universal Approximation Theorem)

5.1 합성곱 신경망의 발달 배경



Single Hidden Layer Perceptron



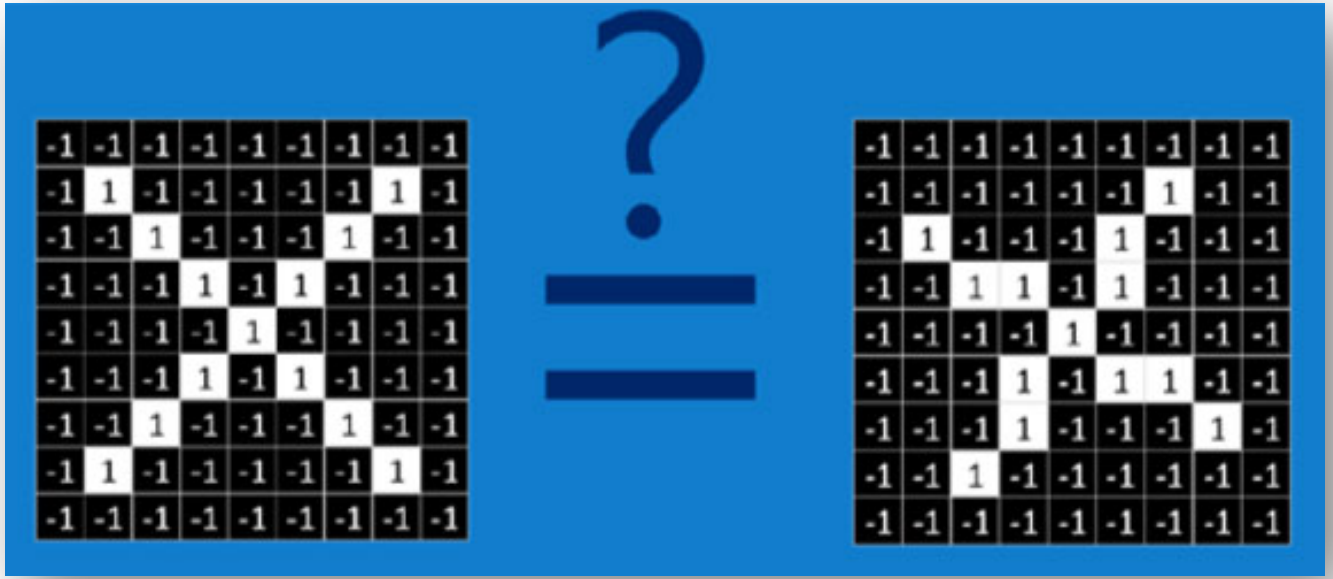
Multi Hidden Layer Perceptron

인공 신경망은 실수 공간에서 연속하는 모든 함수를 근사 가능

Universal Approximation Theorem 실험

5.1 합성곱 신경망의 발달 배경

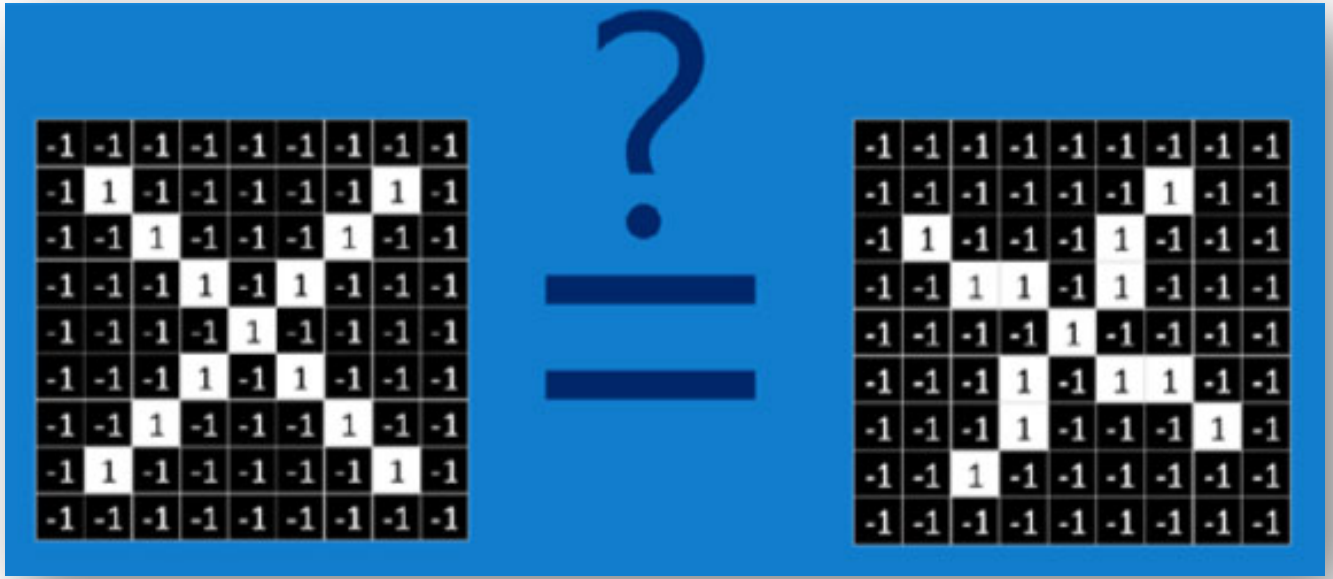
인공 신경망만으로 이미지를 분류할 수 있을까?



How do Convolutional Neural Networks work?

5.1 합성곱 신경망의 발달 배경

인공 신경망만으로 이미지를 분류할 수 있을까?



같은 위치의 픽셀 값이 서로 일치하지 않으면, 다른 이미지로 인식

How do Convolutional Neural Networks work?

5.1 합성곱 신경망의 발달 배경

동물의 시각 뉴런 연구와 CNN의 탄생

- Kunihiko Fukushima
- Yann LeCun

단순 세포

국소적인 영역을 보고
단순한 패턴에 자극을 받음.

복잡 세포

넓은 영역을 보고
복잡한 패턴에 자극을 받음.

동물의 시각 체계는 단순 세포와 복잡 세포의 레이어로 이루어짐

5.1 합성곱 신경망의 발달 배경

동물의 시각 뉴런 연구와 CNN의 탄생

- Kunihiko Fukushima
- Yann LeCun

단순 세포

국소적인 영역을 보고
단순한 패턴에 자극을 받음.

복잡 세포

넓은 영역을 보고
복잡한 패턴에 자극을 받음.

즉, 점이 아닌 영역을 보고, 패턴을 인식하는 과정의 연속

5.1 합성곱 신경망의 발달 배경

LeNet-5의 구조

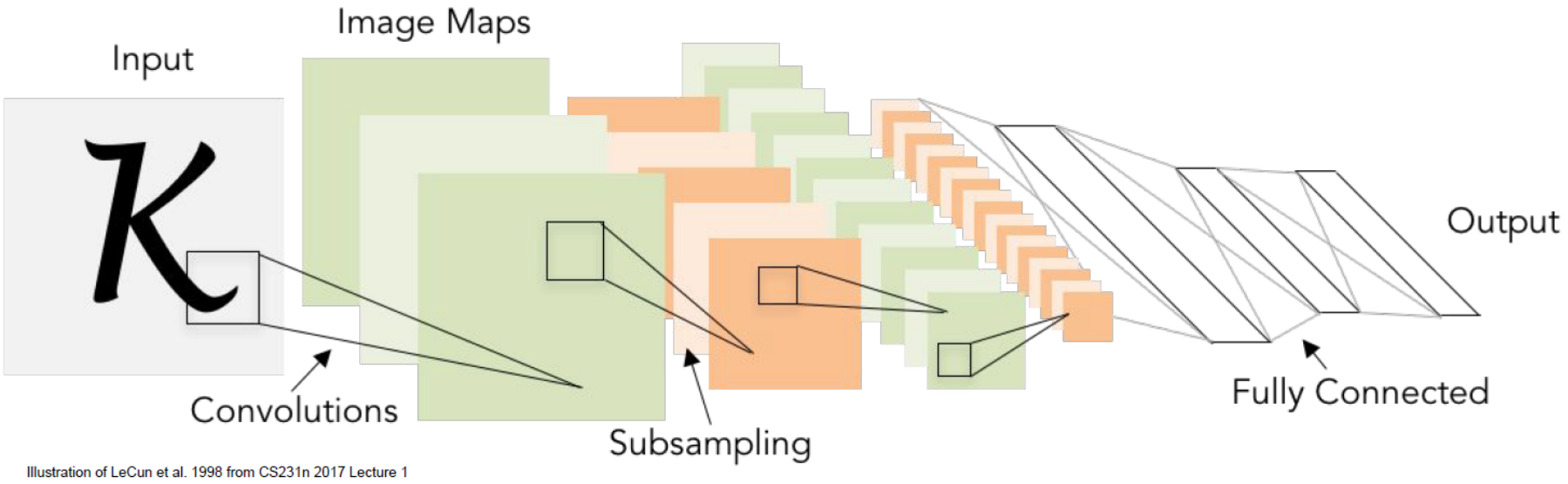
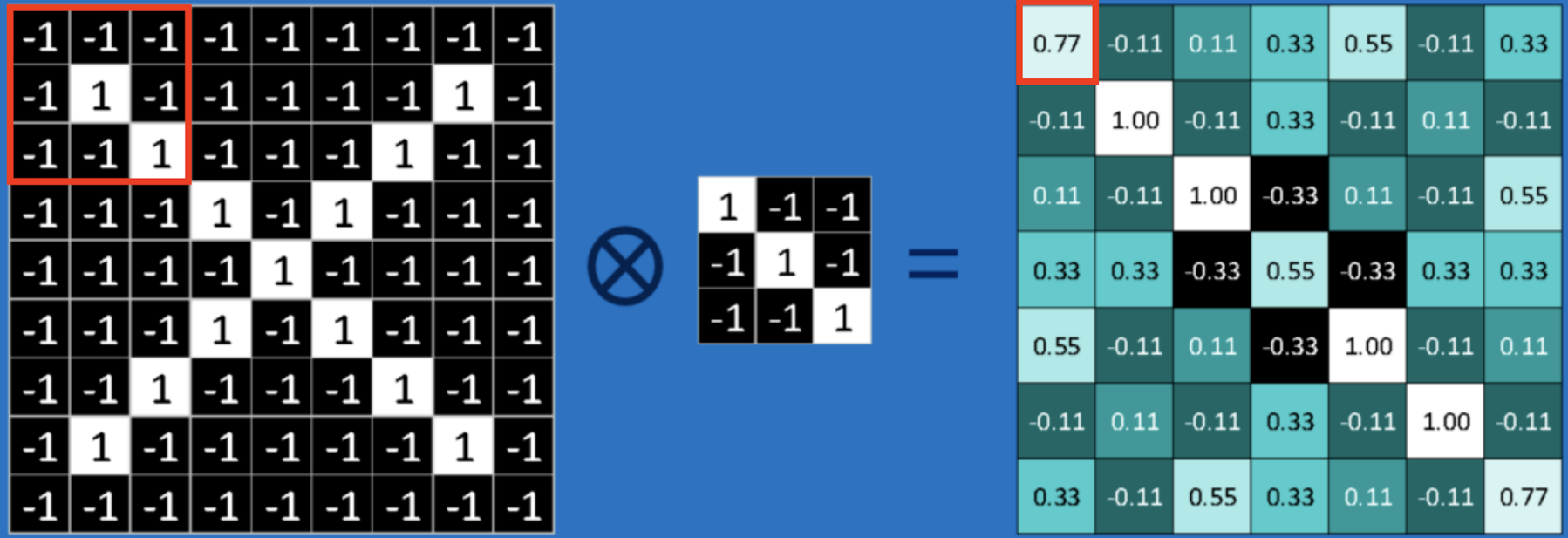


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]

CS231n(2017)

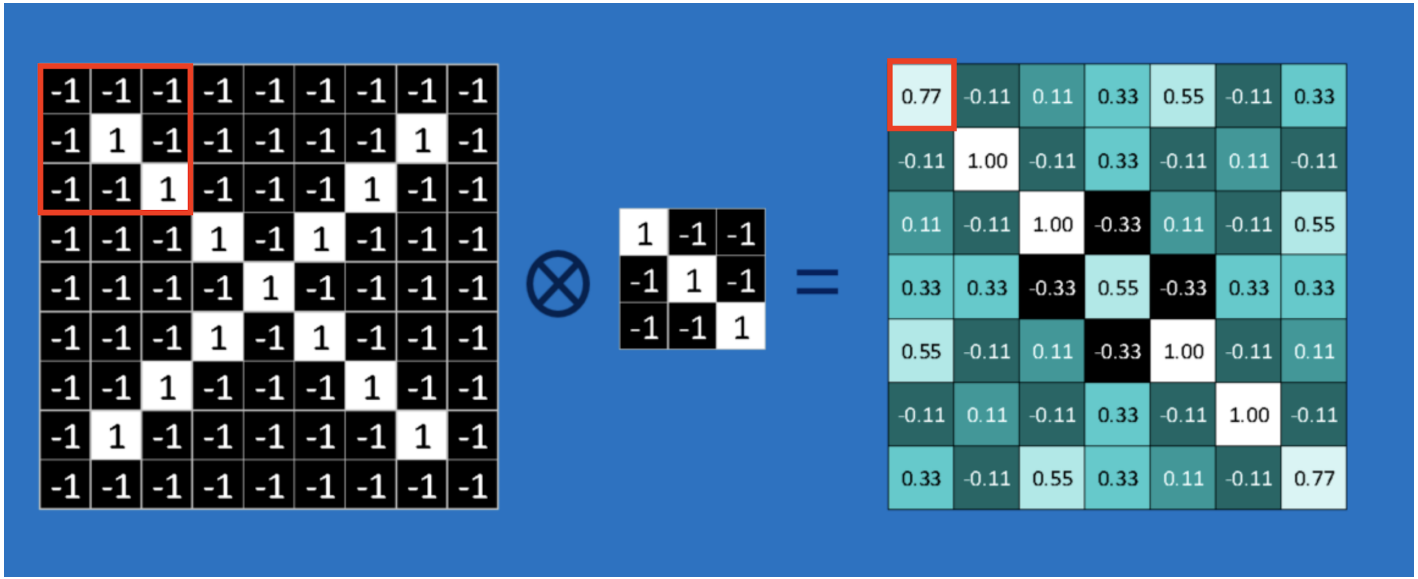
5.2 합성곱 연산 과정



How do Convolutional Neural Networks work?

5.2 합성곱 연산 과정

이미지 필터 피처 맵



$$O = floor(\frac{I - F}{S} + 1)$$

O : $Output\ Size$

I : $Input\ Size$

F : $Filter\ Size$

S : $Stride$

9 x 9
Input Size = 9

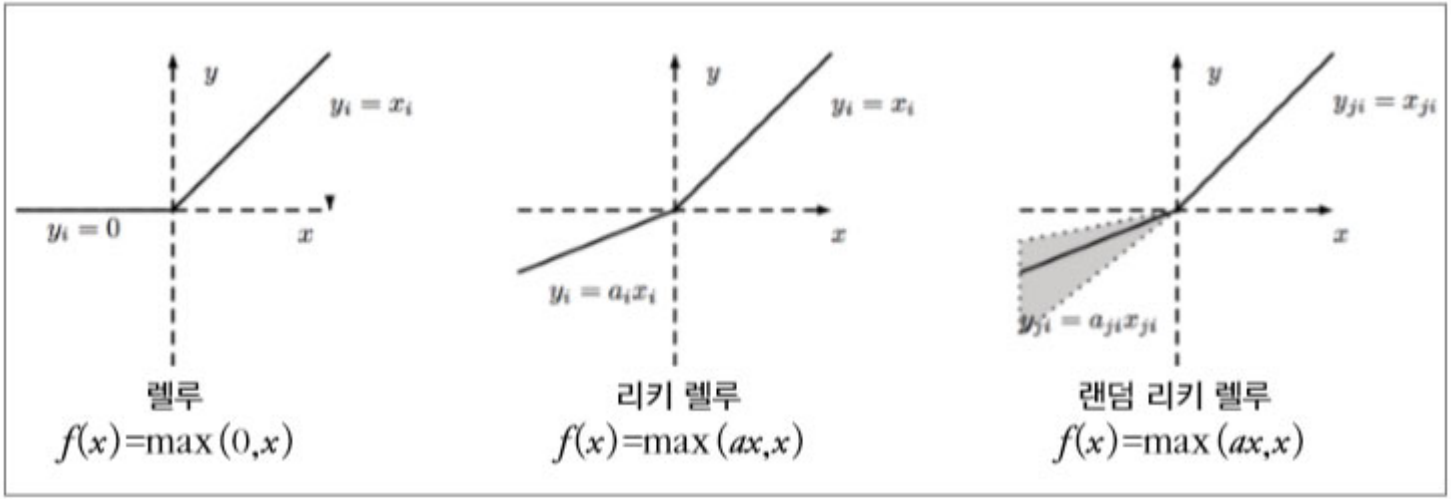
3 x 3
Filter Size = 3

7 x 7
Output Size = 7

How do Convolutional Neural Networks work?

5.2 합성곱 연산 과정

다양한 ReLU 함수

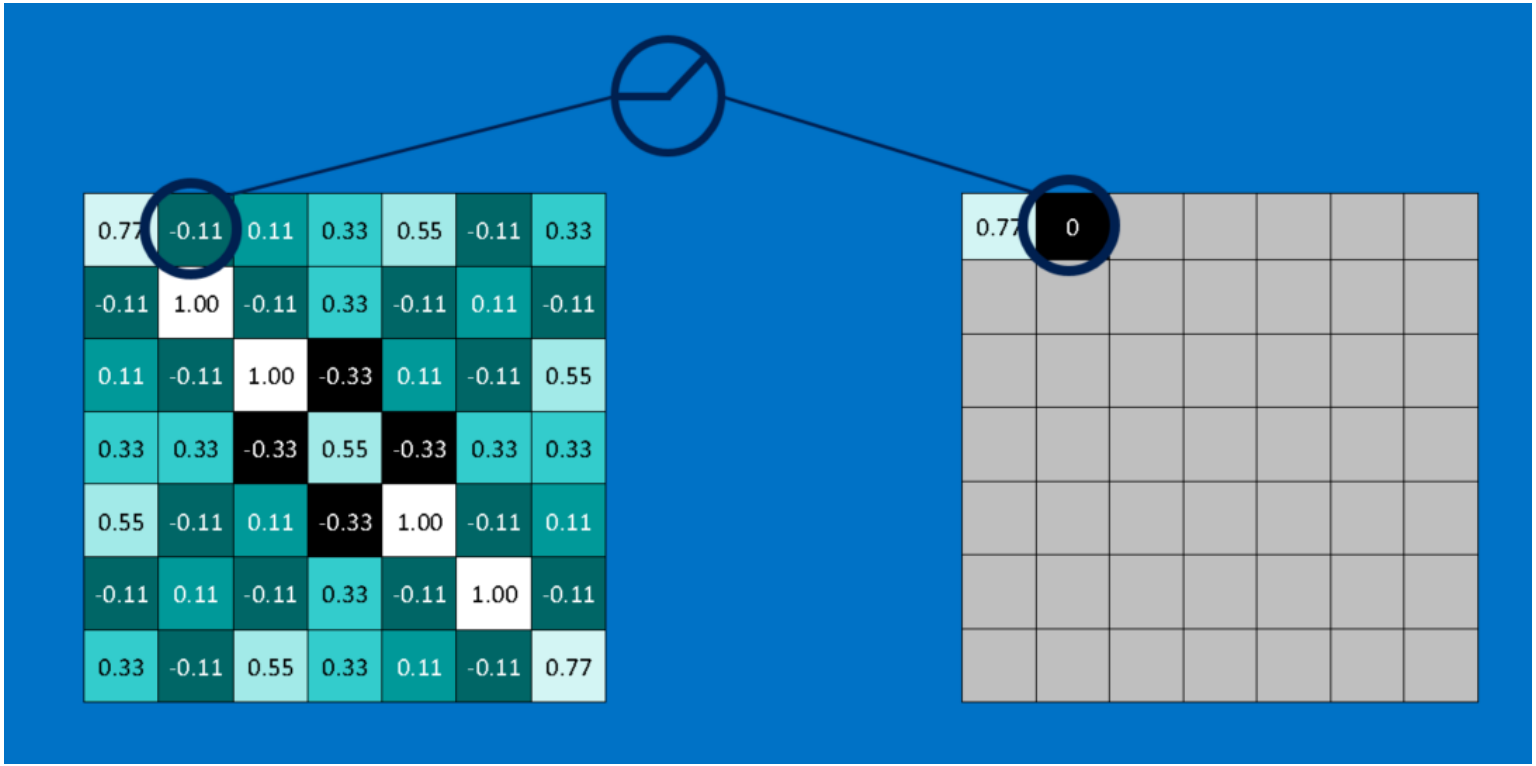


***다잉뉴런(Dying Neuron)** : 어떠한 입력값이 들어와도 활성화 값이 0이 되는 현상.

예) 입력값(X)이 **−1 ~ 1** 사이로 정규화되어 있고, **W = −3, b = −5**일 경우
WX+b는 **−2 ~ −8**로 음수값을 가지게 되어, 활성화 값은 **0**이 됨.

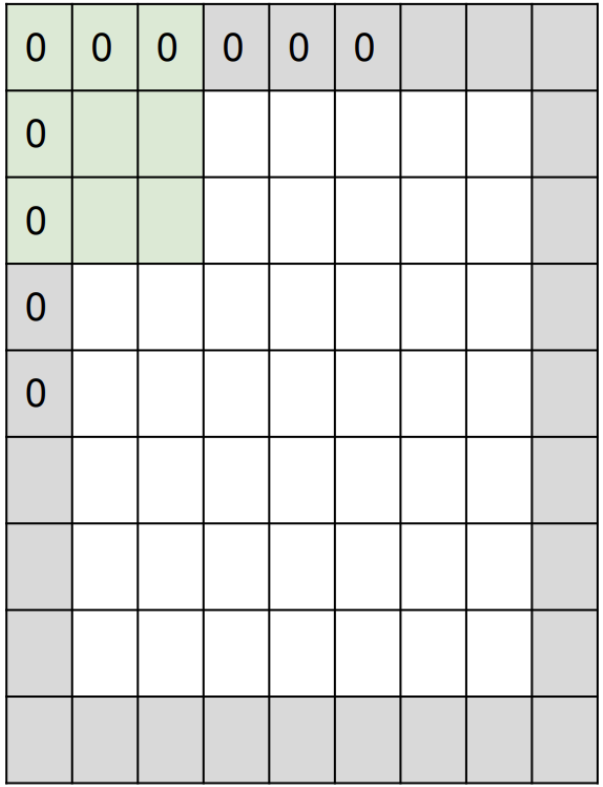
5.2 합성곱 연산 과정

피처 맵에 ReLU 함수적용



How do Convolutional Neural Networks work?

5.3 패딩과 풀링



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

$$O = floor(\frac{I - F}{S} + 1)$$

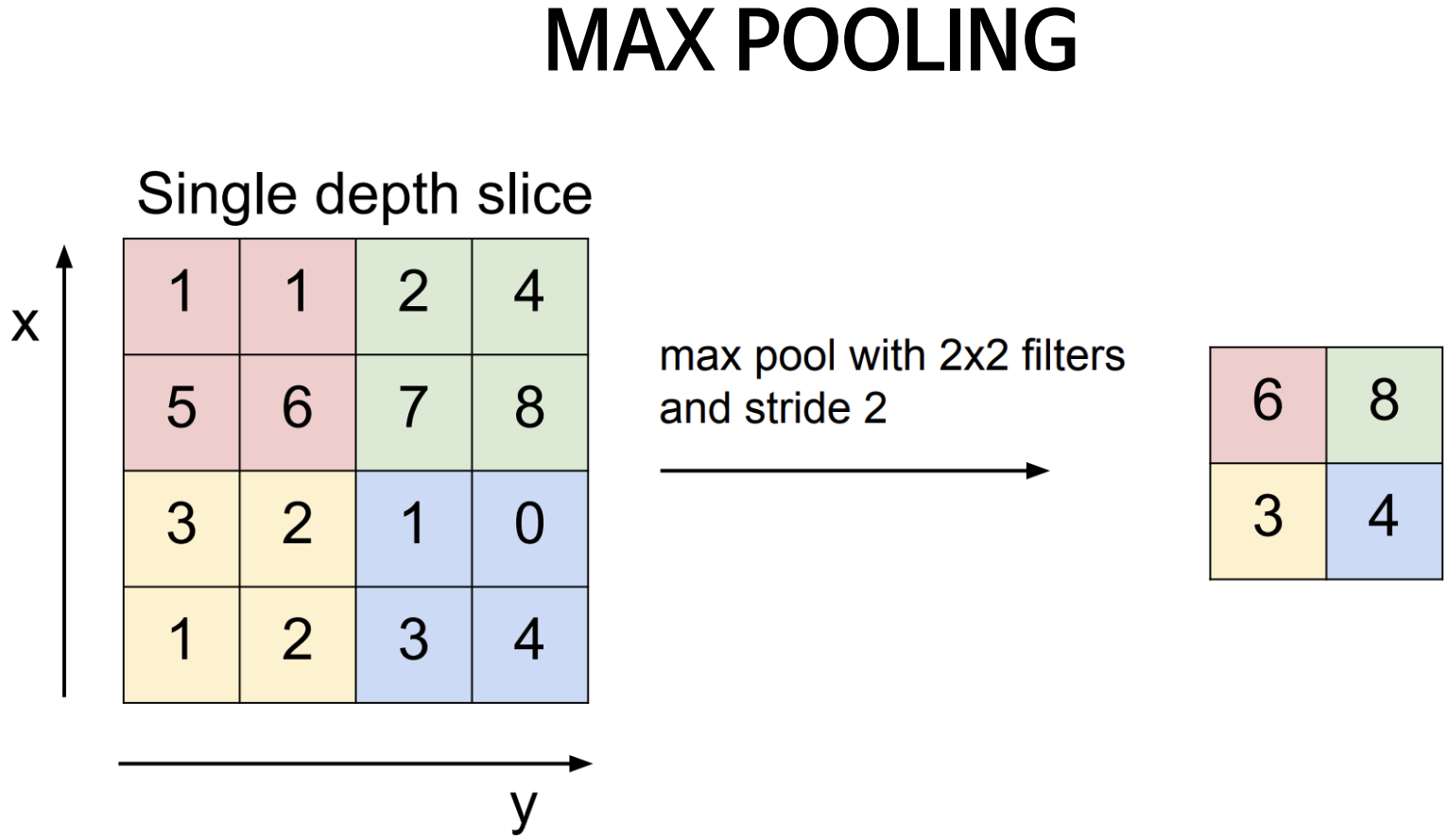
O : $Output\ Size$
I : $Input\ Size$
F : $Filter\ Size$
S : $Stride$

$$O = floor(\frac{I - F + 2P}{S} + 1)$$

O : $Output\ Size$
I : $Input\ Size$
F : $Filter\ Size$
P : $Padding\ Size$
S : $Stride$

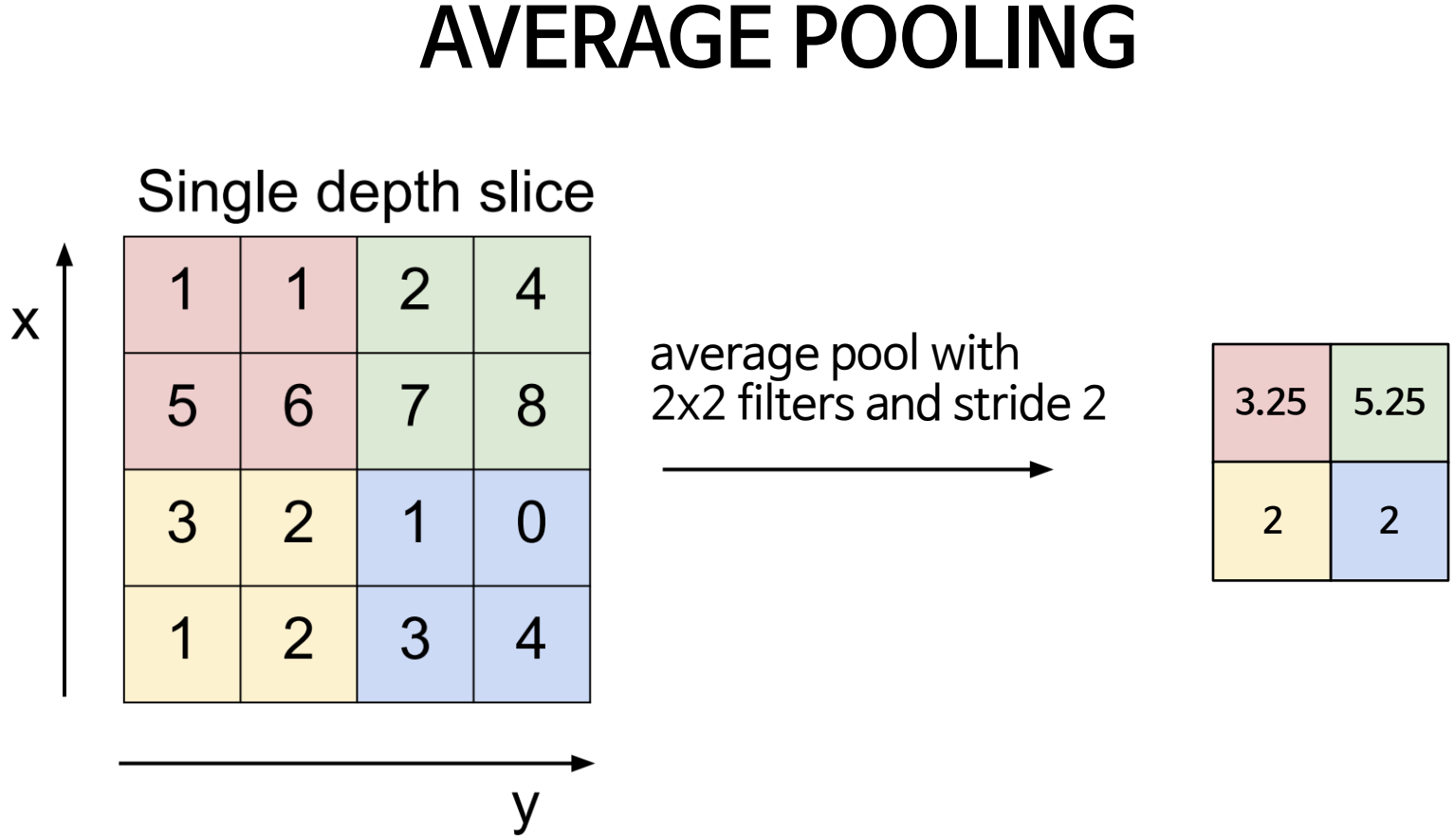
CS231n(2017)

5.3 패딩과 풀링

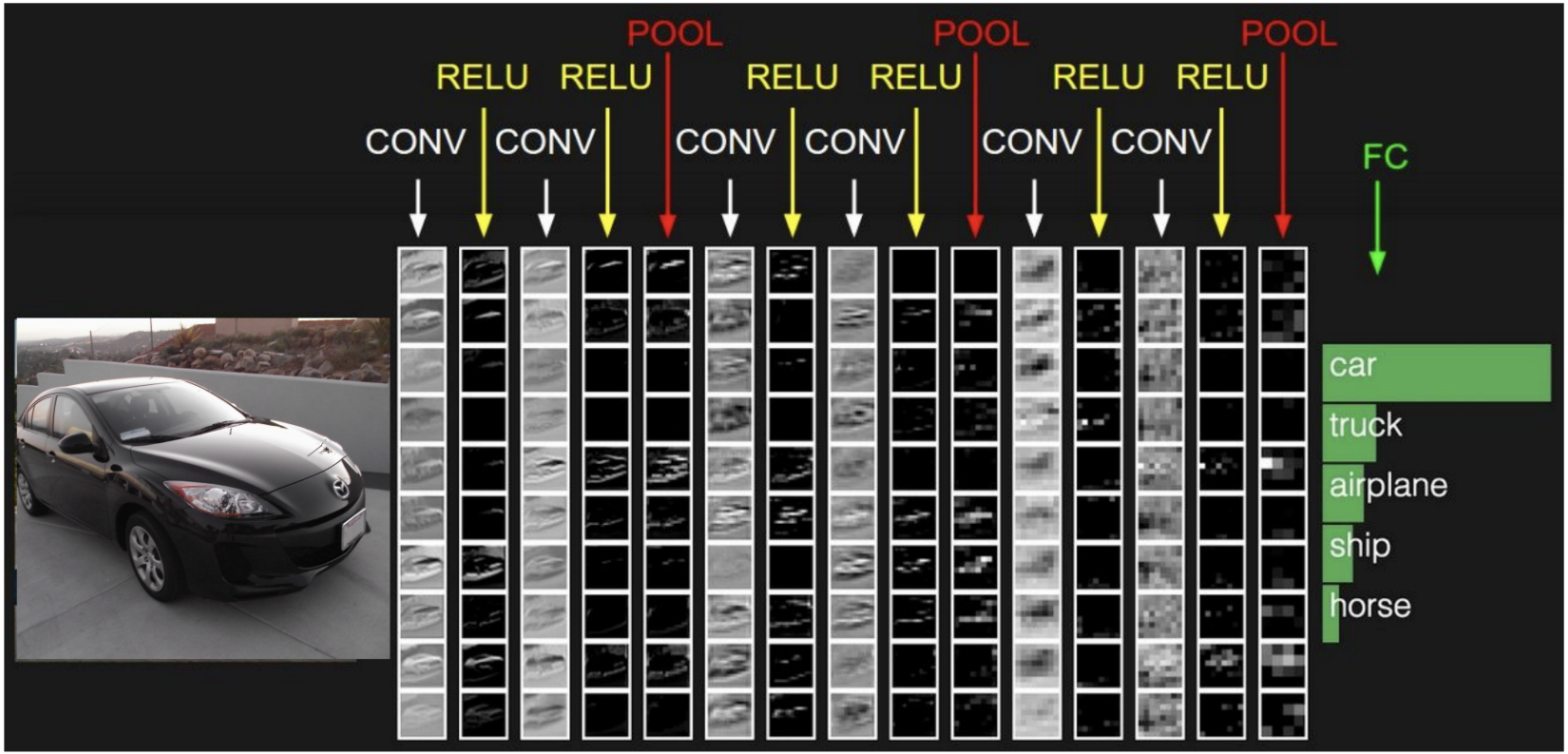


CS231n(2017)

5.3 패딩과 풀링

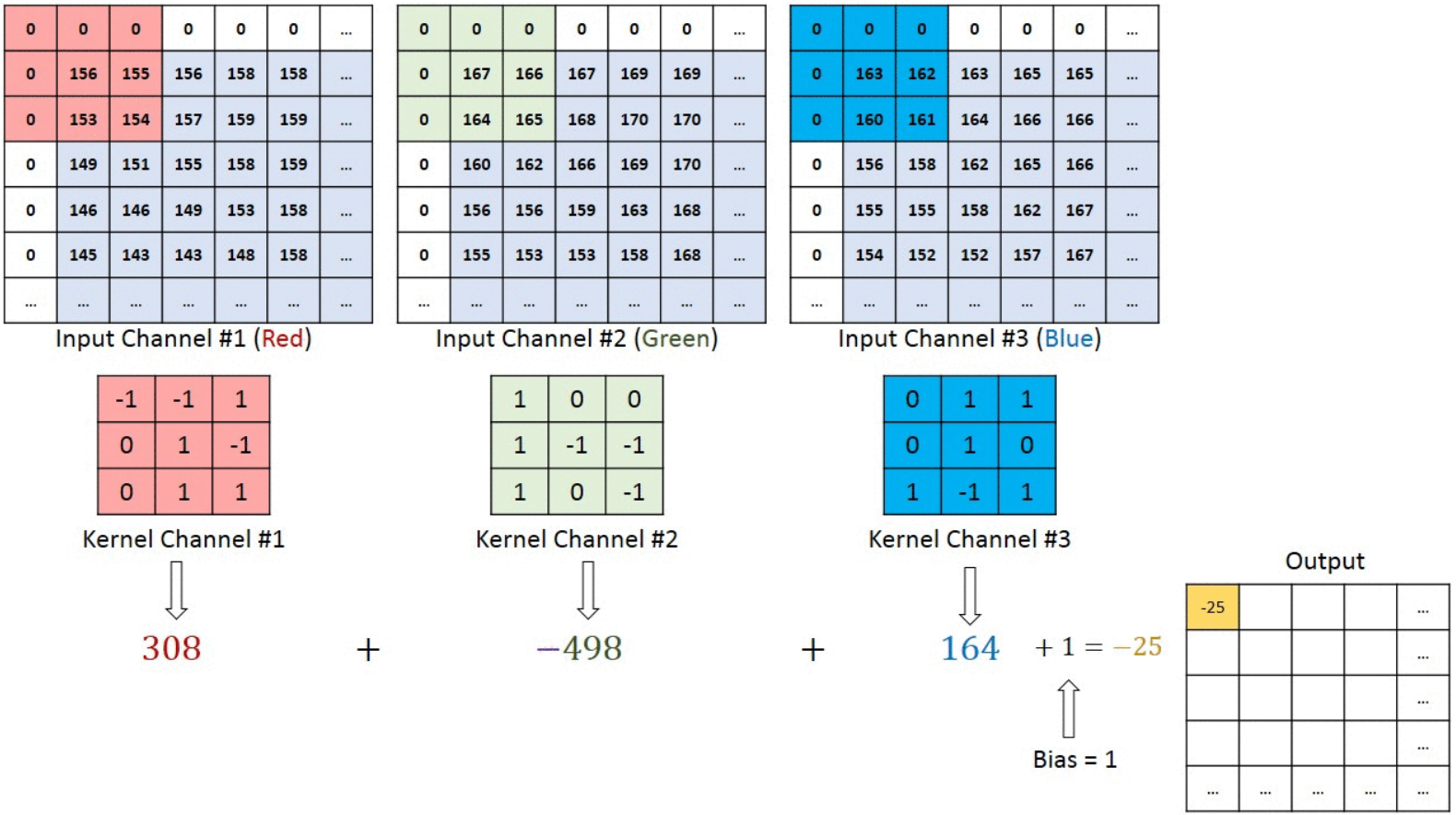

AVERAGE POOLING
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
average pool with
2x2 filters and stride 2
3.25 5.25
2 2

5.3 패딩과 풀링

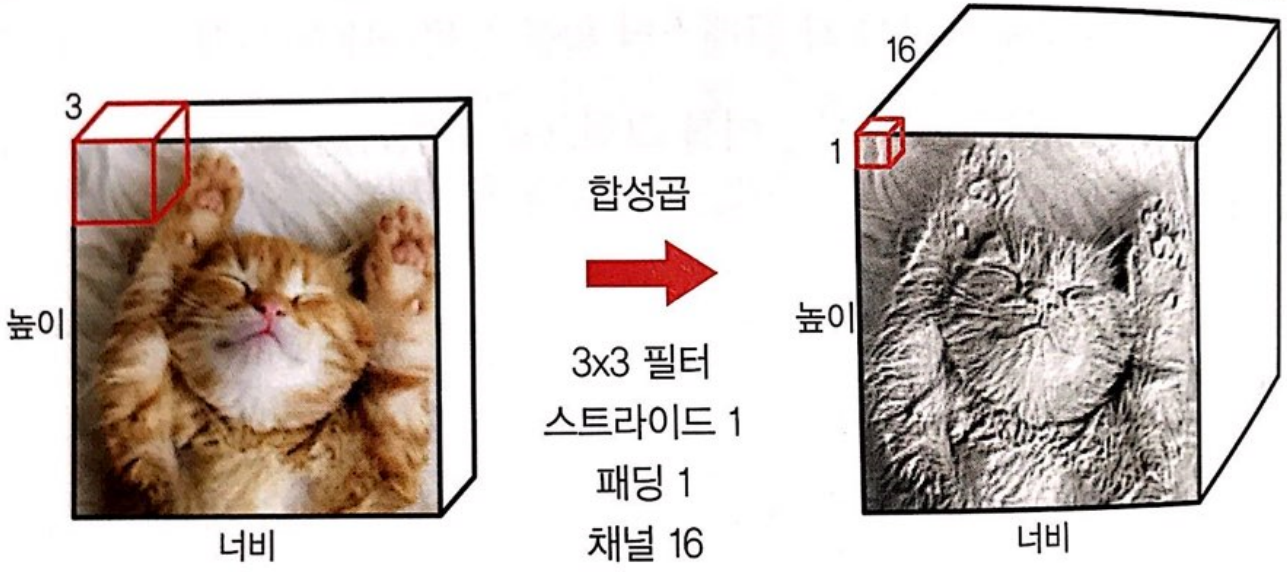


CS231n(2017)

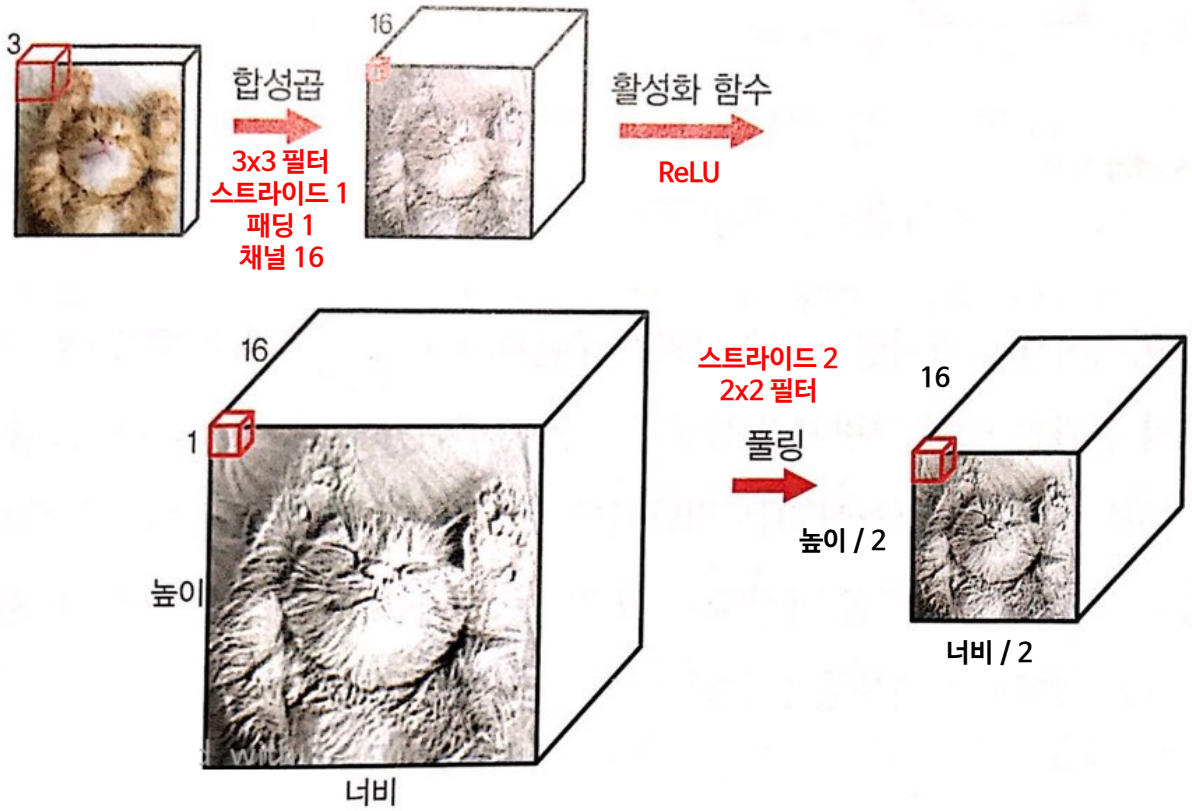
5.4 모델의 3차원적 이해



5.4 모델의 3차원적 이해

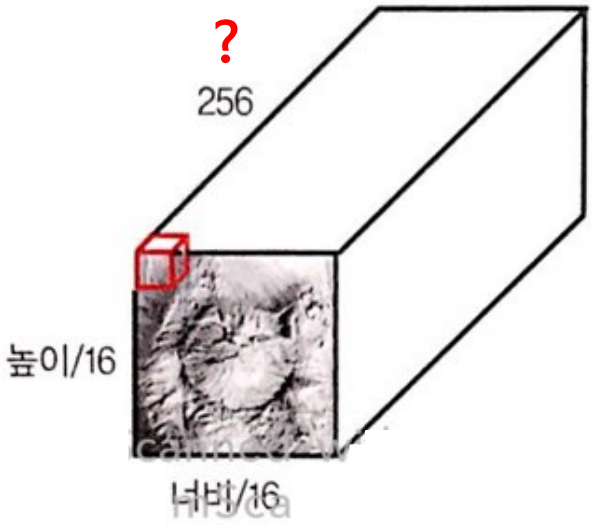


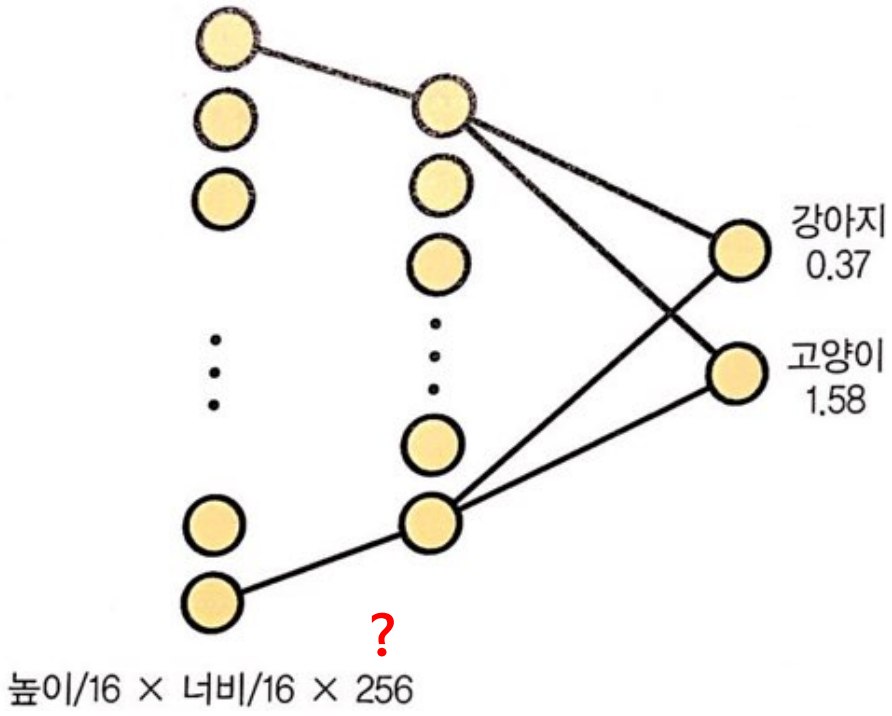
5.4 모델의 3차원적 이해



5.4 모델의 3차원적 이해

Q. 풀링 4회 수행 후 채널이 증가한 이유?





5.5 소프트맥스 함수

One-Hot Encoding

Target
Dog
Cat

변환

Dog	Cat
1	0
0	1

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}$$

$$n : Number\ of\ classes$$

$$i : i\ class$$

Softmax Function

Feed Forward Output

Dog	Cat
2	7

Softmax Output

Dog	Cat
0.01	0.99

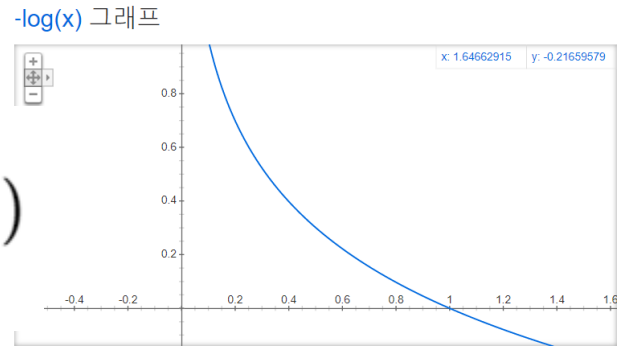
```
import numpy as np

y = np.array([2,7])
exp_y = np.exp(y)
exp_y/exp_y.sum()
```

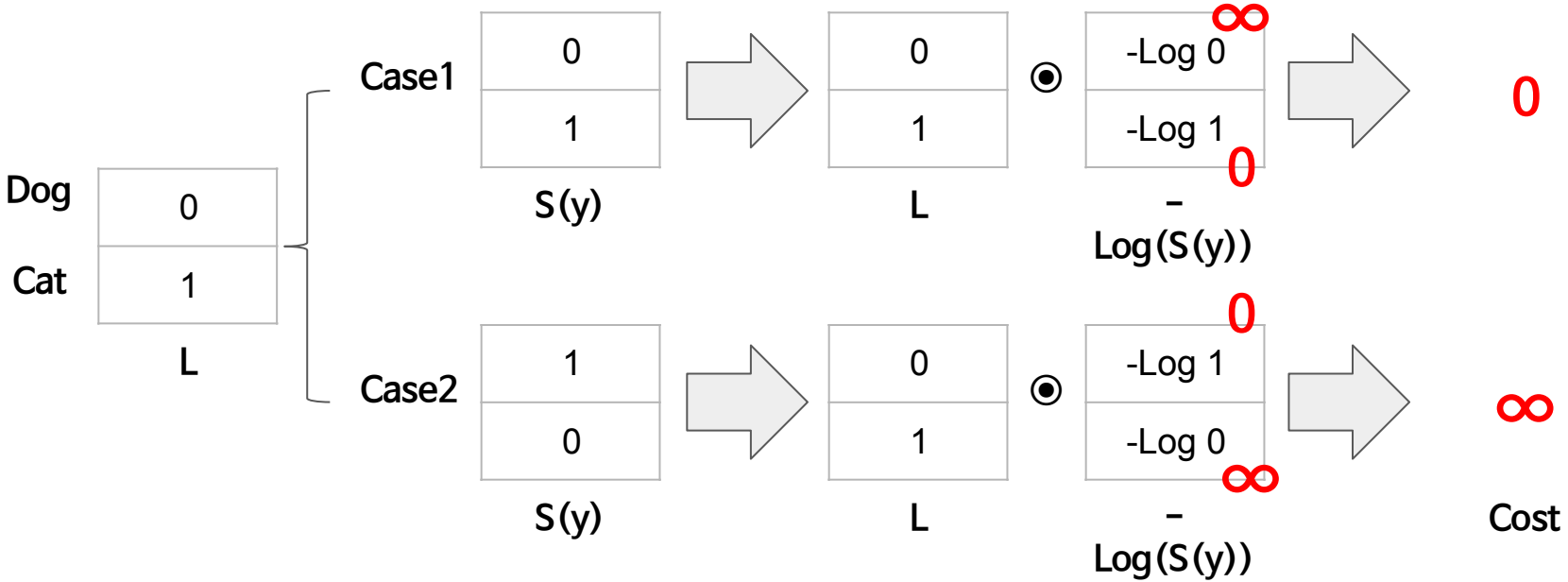
```
array([0.00669285, 0.99330715])
```

5.5 소프트맥스 함수

Cross Entropy

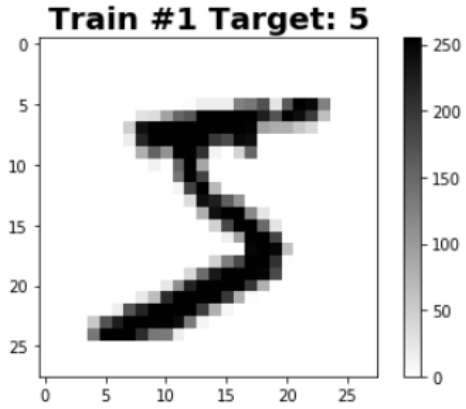
$$D(S, L) = -\sum_{j=1}^{n} L_j log(S(y_j))$$





5.6 모델 구현, 학습 및 결과 확인

Code



Input Image

28 x 28 x 1

5 x 5 필터
스트라이드 1
패딩 0
채널 16

Conv
ReLU

24 x 24 x 16

5 x 5 필터
스트라이드 1
패딩 0
채널 32

Conv
ReLU

20 x 20 x 32

2 x 2 필터
스트라이드 2
패딩 0

Max Pool

10 x 10 x 32

5 x 5 필터
스트라이드 1
패딩 0
채널 64

Conv
ReLU

6 x 6 x 64

2 x 2 필터
스트라이드 1
패딩 0

Max Pool

3 x 3 x 64

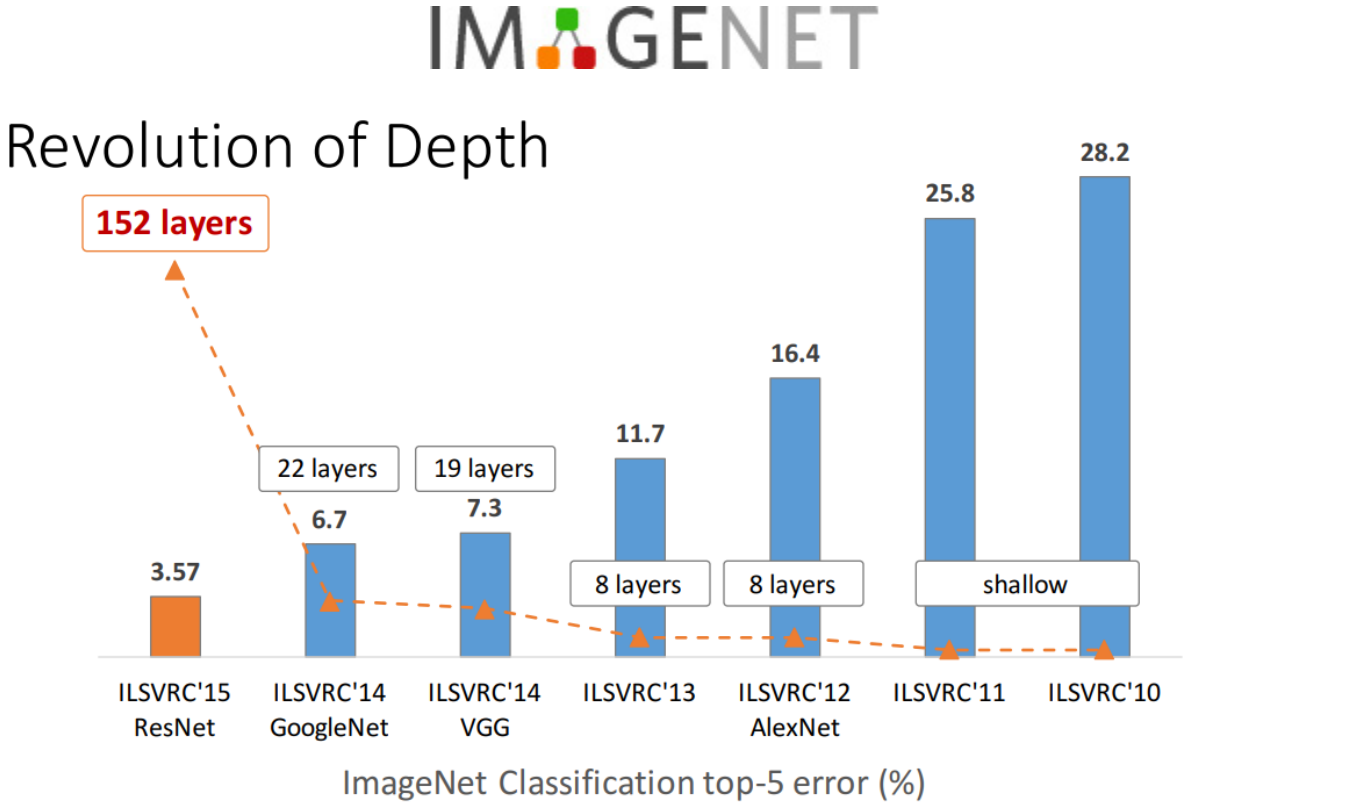
Fully Connected

3 x 3 x 64
Input

100
Hidden Layer

10
Output Layer

5.7 유명한 모델들과 원리



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

5.7 유명한 모델들과 원리

Code

VGGNet

https://arxiv.org/pdf/1409.1556.pdf

*2014 ILSVRC 2nd place

3x3 Conv, 2x2 MaxPool, 1 Padding, 1 Stride

5.7 유명한 모델들과 원리

Code

VGGNet

https://arxiv.org/pdf/1409.1556.pdf

*2014 ILSVRC 2nd place

즉, Pooling 연산 후에만 이미지 사이즈가 반으로 줄어듦

5.7 유명한 모델들과 원리

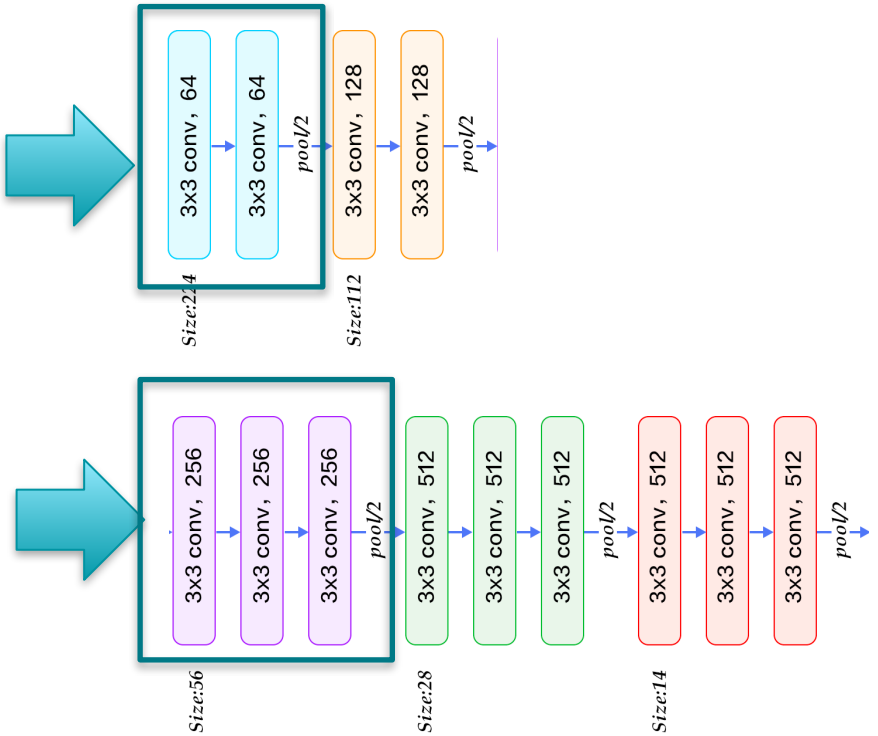
Code

VGGNet

https://arxiv.org/pdf/1409.1556.pdf

*2014 ILSVRC 2nd place

```
# 컨볼루션 연산이 2번 연속하는 경우
# 컨볼루션-활성화함수-컨볼루션-활성화함수-풀링
def conv_2_block(in_dim,out_dim):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim,kernel_size=3,padding=1),
        nn.ReLU(),
        nn.Conv2d(out_dim,out_dim,kernel_size=3,padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2,2)
    )
    return model


# 컨볼루션 연산이 3번 연속하는 경우
# 컨볼루션-활성화함수-컨볼루션-활성화함수-컨볼루션-활성화함수-풀링
def conv_3_block(in_dim,out_dim):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim,kernel_size=3,padding=1),
        nn.ReLU(),
        nn.Conv2d(out_dim,out_dim,kernel_size=3,padding=1),
        nn.ReLU(),
        nn.Conv2d(out_dim,out_dim,kernel_size=3,padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2,2)
    )
    return model
```



5.7 유명한 모델들과 원리

Code

VGGNet

https://arxiv.org/pdf/1409.1556.pdf

*2014 ILSVRC 2nd place

```
class VGG(nn.Module):
    def __init__(self, base_dim, num_classes=2):
        super(VGG, self).__init__()
        self.feature = nn.Sequential(
            conv_2_block(3,base_dim),
            conv_2_block(base_dim,2*base_dim),
            conv_3_block(2*base_dim,4*base_dim),
            conv_3_block(4*base_dim,8*base_dim),
            conv_3_block(8*base_dim,8*base_dim),
        )
        self.fc_layer = nn.Sequential(
            nn.Linear(8*base_dim * 7 * 7, 100),
            nn.ReLU(True),
            #nn.Dropout(),
            nn.Linear(100, 20),
            nn.ReLU(True),
            #nn.Dropout(),
            nn.Linear(20, num_classes),
        )

    def forward(self, x):
        x = self.feature(x)
        x = x.view(x.size(0), -1)
        x = self.fc_layer(x)
        return x
```

입력 이미지 RGB 채널 = 3

Conv 채널 = 64

Conv 채널 = 512

최종 이미지 사이즈 = 7 x 7

FC Hidden Layer 1 노드 수 = 100

FC Hidden Layer 2 노드 수 = 20

Output Layer 노드 수 = 2 (dog or cat)

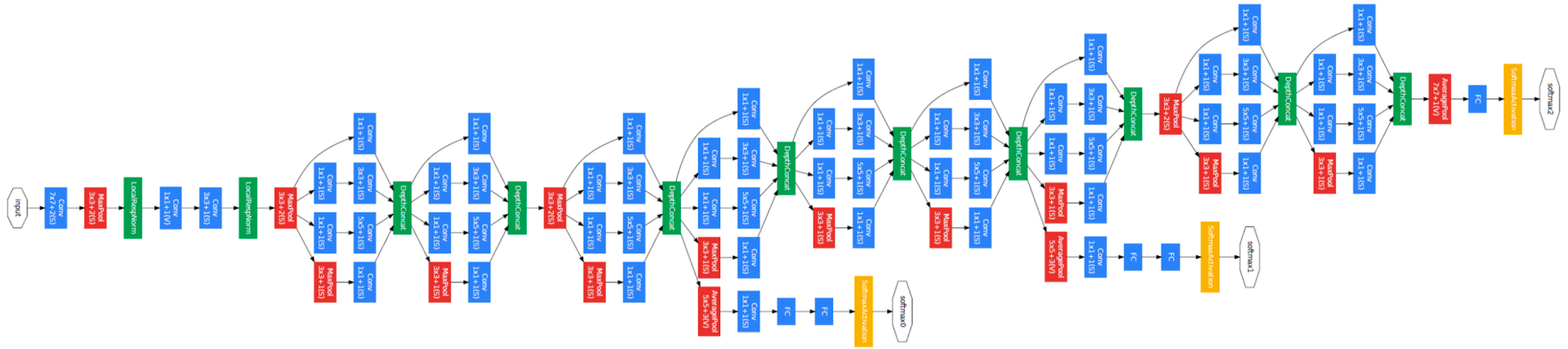
5.7 유명한 모델들과 원리

Code

GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place



A.k.a 인셉션 네트워크

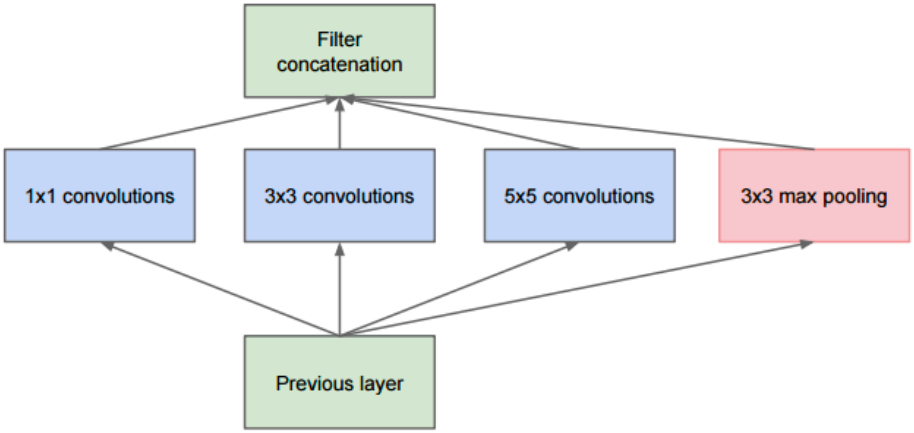
5.7 유명한 모델들과 원리

Code

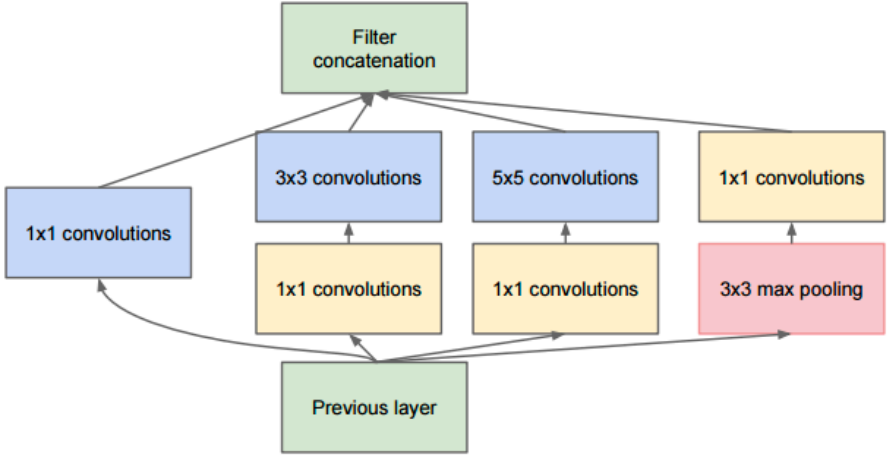
GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place



(a) Inception module, naïve version



(b) Inception module with dimension reductions

1x1 Conv를 추가하면 메모리 효율성이 높아짐

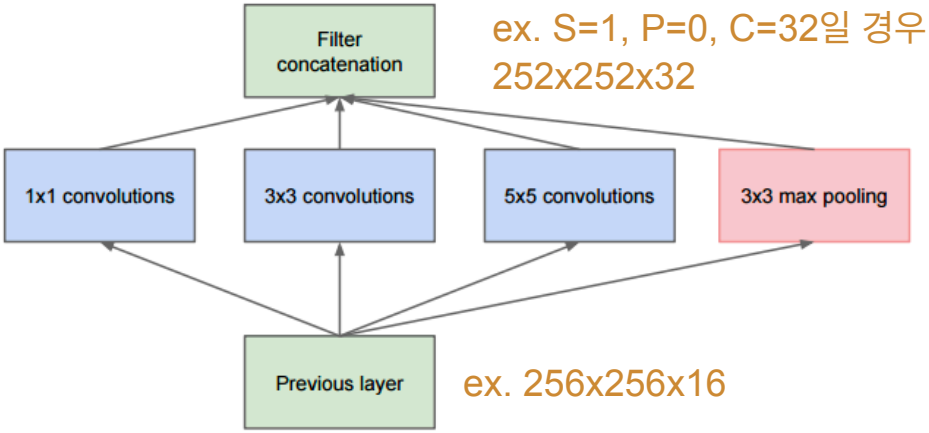
5.7 유명한 모델들과 원리

Code

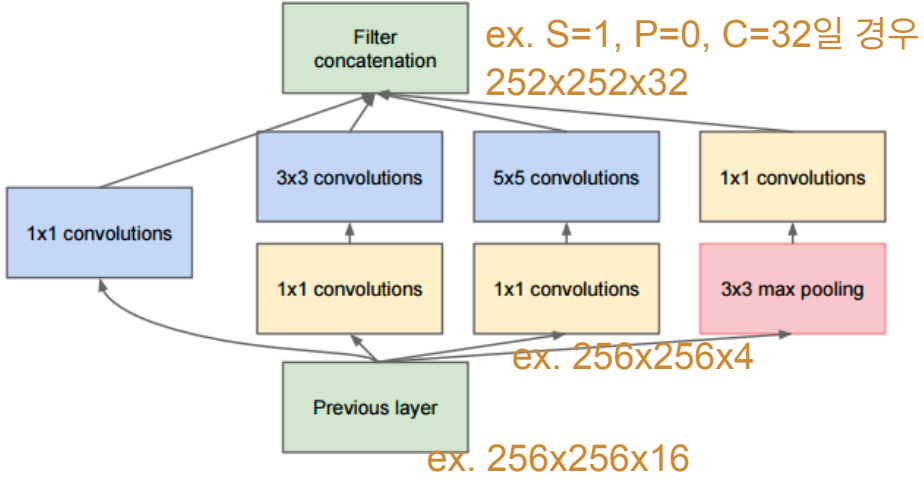
GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place



(a) Inception module, naïve version



(b) Inception module with dimension reductions

직전 텐서의 가로, 세로 사이즈는 유지하면서 채널 수를 줄일 수 있음

5.7 유명한 모델들과 원리

Code

GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

5.7 유명한 모델들과 원리

Code

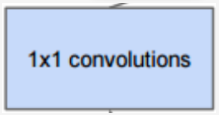
GoogLeNet

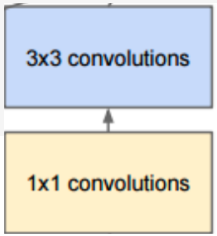
https://arxiv.org/pdf/1409.4842.pdf

커널(필터)=1, 스트라이드=1, (패딩=0)

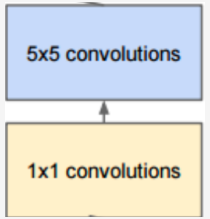
*2014 ILSVRC 1st place

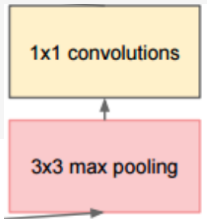
```
def conv_1(in_dim,out_dim):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim,1,1),
        nn.ReLU(),
    )
    return model

def conv_1_3(in_dim,mid_dim,out_dim):
    model = nn.Sequential(
        nn.Conv2d(in_dim,mid_dim,1,1),
        nn.ReLU(),
        nn.Conv2d(mid_dim,out_dim,3,1,1),
        nn.ReLU()
    )
    return model
```






```
def conv_1_5(in_dim,mid_dim,out_dim):
    model = nn.Sequential(
        nn.Conv2d(in_dim,mid_dim,1,1),
        nn.ReLU(),
        nn.Conv2d(mid_dim,out_dim,5,1,2),
        nn.ReLU()
    )
    return model

def max_3_1(in_dim,out_dim):
    model = nn.Sequential(
        nn.MaxPool2d(3,1,1),
        nn.Conv2d(in_dim,out_dim,1,1),
        nn.ReLU(),
    )
    return model
```





5.7 유명한 모델들과 원리

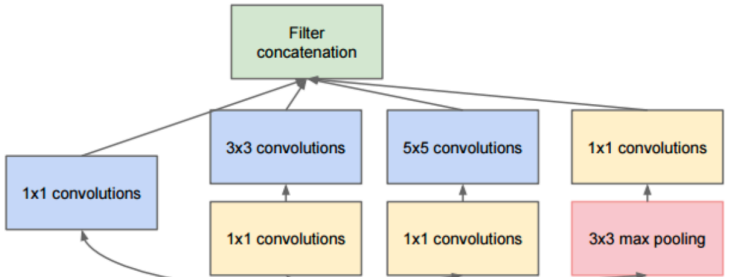
Code

GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place

```
class inception_module(nn.Module):
    def __init__(self,in_dim,out_dim_1,mid_dim_3,out_dim_3,mid_dim_5,out_dim_5,pool):
        super(inception_module,self).__init__()
        self.conv_1 = conv_1(in_dim,out_dim_1)
        self.conv_1_3 = conv_1_3(in_dim,mid_dim_3,out_dim_3)
        self.conv_1_5 = conv_1_5(in_dim,mid_dim_5,out_dim_5)
        self.max_3_1 = max_3_1(in_dim,pool)

    def forward(self,x):
        out_1 = self.conv_1(x)
        out_2 = self.conv_1_3(x)
        out_3 = self.conv_1_5(x)
        out_4 = self.max_3_1(x)
        output = torch.cat([out_1,out_2,out_3,out_4],1)
        return output
```



5.7 유명한 모델들과 원리

Code

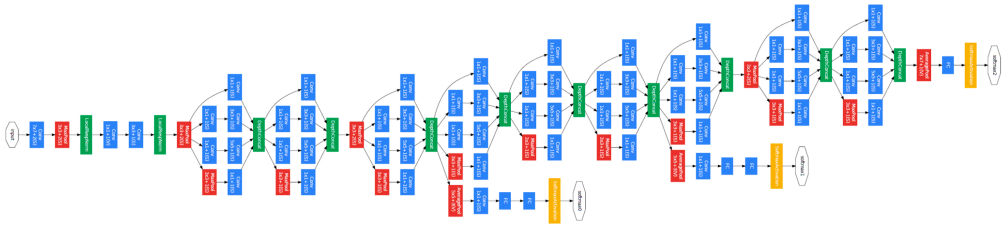
GoogLeNet

https://arxiv.org/pdf/1409.4842.pdf

*2014 ILSVRC 1st place

```
class GoogLeNet(nn.Module):
    def __init__(self, base_dim, num_classes=2):
        super(GoogLeNet, self).__init__()
        self.num_classes=num_classes
        self.layer_1 = nn.Sequential(
            nn.Conv2d(3,base_dim,7,2,3),
            nn.MaxPool2d(3,2,1),
            nn.Conv2d(base_dim,base_dim*3,3,1,1),
            nn.MaxPool2d(3,2,1),
        )
        self.layer_2 = nn.Sequential(
            inception_module(base_dim*3,64,96,128,16,32,32),
            inception_module(base_dim*4,128,128,192,32,96,64),
            nn.MaxPool2d(3,2,1),
        )
        self.layer_3 = nn.Sequential(
            inception_module(480,192,96,208,16,48,64),
            inception_module(512,160,112,224,24,64,64),
            inception_module(512,128,128,256,24,64,64),
            inception_module(512,112,144,288,32,64,64),
            inception_module(528,256,160,320,32,128,128),
            nn.MaxPool2d(3,2,1),
        )
        self.layer_4 = nn.Sequential(
            inception_module(832,256,160,320,32,128,128),
            inception_module(832,384,192,384,48,128,128),
            nn.AvgPool2d(7,1),
        )
        self.layer_5 = nn.Dropout2d(0.4)
        self.fc_layer = nn.Linear(1024,self.num_classes)
```

```
def forward(self, x):
    out = self.layer_1(x)
    out = self.layer_2(out)
    out = self.layer_3(out)
    out = self.layer_4(out)
    out = self.layer_5(out)
    out = out.view(batch_size,-1)
    out = self.fc_layer(out)
    return out
```



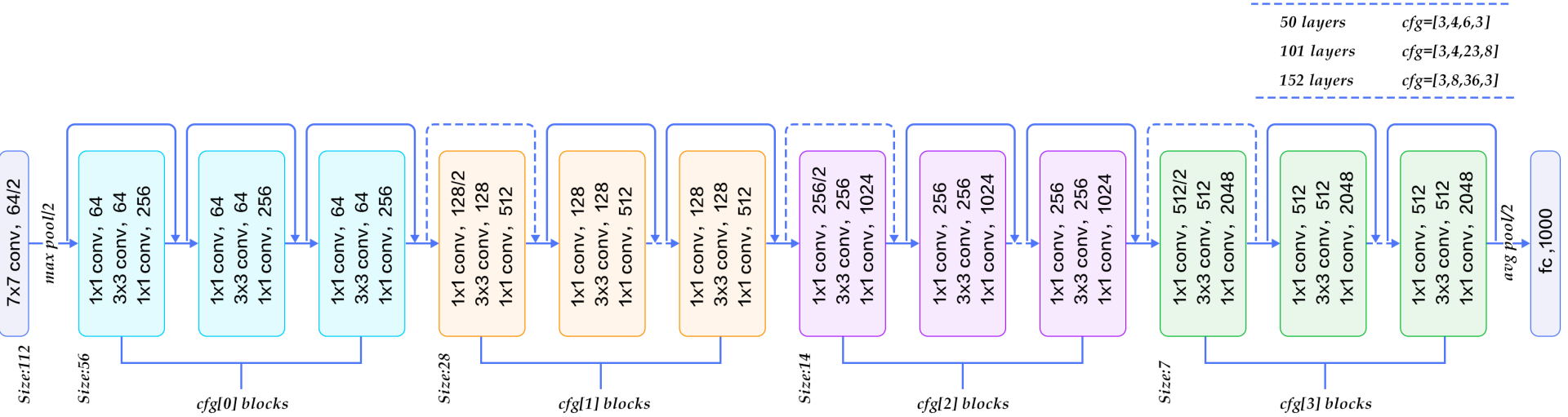
5.7 유명한 모델들과 원리

Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place



잔차 학습을 통한 매우 깊은 네트워크 제시

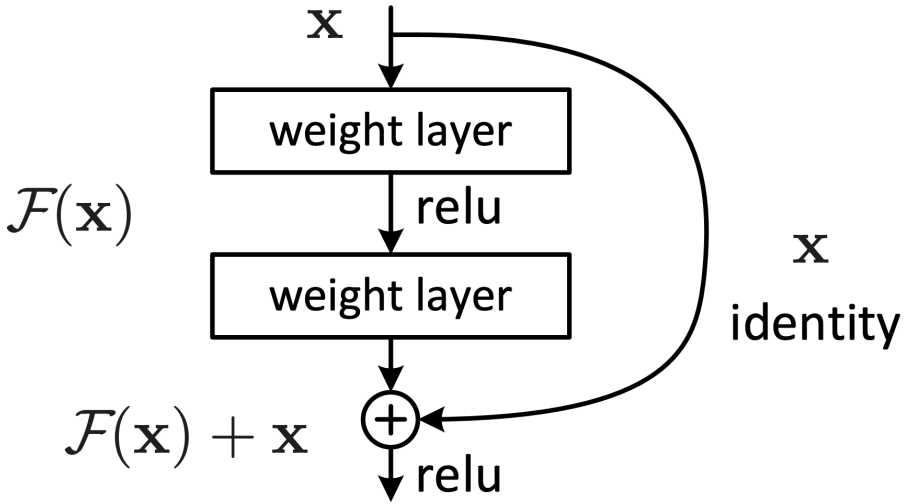
5.7 유명한 모델들과 원리

Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place



Skip Connection

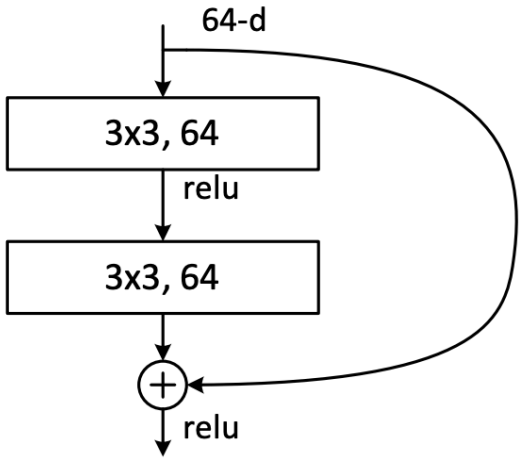
5.7 유명한 모델들과 원리

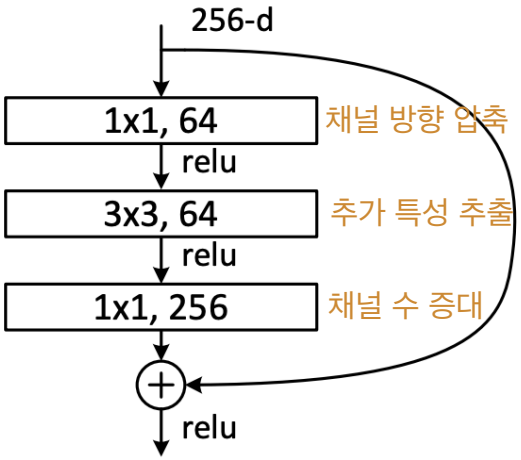
Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place





Bottleneck Block

5.7 유명한 모델들과 원리

Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place

```
def conv_block_1(in_dim,out_dim,act_fn,stride=1):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim, kernel_size=1, stride=stride),
        act_fn,
    )
    return model


def conv_block_3(in_dim,out_dim,act_fn):
    model = nn.Sequential(
        nn.Conv2d(in_dim,out_dim, kernel_size=3, stride=1, padding=1),
        act_fn,
    )
    return model
```

5.7 유명한 모델들과 원리

Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place

```
class BottleNeck(nn.Module):
    def __init__(self,in_dim,mid_dim,out_dim,act_fn,down=False):
        super(BottleNeck,self).__init__()
        self.down=down

        # 특성지도의 크기가 감소하는 경우
        if self.down:
            self.layer = nn.Sequential(
              conv_block_1(in_dim,mid_dim,act_fn,2),
              conv_block_3(mid_dim,mid_dim,act_fn),
              conv_block_1(mid_dim,out_dim,act_fn),
            )
            self.downsample = nn.Conv2d(in_dim,out_dim,1,2)

        # 특성지도의 크기가 그대로인 경우
        else:
            self.layer = nn.Sequential(
                conv_block_1(in_dim,mid_dim,act_fn),
                conv_block_3(mid_dim,mid_dim,act_fn),
                conv_block_1(mid_dim,out_dim,act_fn),
            )

        # 더하기를 위해 차원을 맞춰주는 부분
        self.dim_equalizer = nn.Conv2d(in_dim,out_dim,kernel_size=1)
```

스트라이드 = 2

```
    def forward(self,x):
        if self.down:
            downsample = self.downsample(x)
            out = self.layer(x)
            out = out + downsample
        else:
            out = self.layer(x)
            if x.size() is not out.size():
                x = self.dim_equalizer(x)
            out = out + x
        return out
```

5.7 유명한 모델들과 원리

Code

ResNet

https://arxiv.org/pdf/1512.03385.pdf

*2015 ILSVRC 1st place

```
class ResNet(nn.Module):

    def __init__(self, base_dim, num_classes=2):
        super(ResNet, self).__init__()
        self.act_fn = nn.ReLU()
        self.layer_1 = nn.Sequential(
            nn.Conv2d(3,base_dim,7,2,3),
            nn.ReLU(),
            nn.MaxPool2d(3,2,1),
        )
        self.layer_2 = nn.Sequential(
            BottleNeck(base_dim,base_dim,base_dim*4,self.act_fn),
            BottleNeck(base_dim*4,base_dim,base_dim*4,self.act_fn),
            BottleNeck(base_dim*4,base_dim,base_dim*4,self.act_fn,down=True),
        )
        self.layer_3 = nn.Sequential(
            BottleNeck(base_dim*4,base_dim*2,base_dim*8,self.act_fn),
            BottleNeck(base_dim*8,base_dim*2,base_dim*8,self.act_fn),
            BottleNeck(base_dim*8,base_dim*2,base_dim*8,self.act_fn),
            BottleNeck(base_dim*8,base_dim*2,base_dim*8,self.act_fn,down=True),
        )
        self.layer_4 = nn.Sequential(
            BottleNeck(base_dim*8,base_dim*4,base_dim*16,self.act_fn),
            BottleNeck(base_dim*16,base_dim*4,base_dim*16,self.act_fn),
            BottleNeck(base_dim*16,base_dim*4,base_dim*16,self.act_fn),
            BottleNeck(base_dim*16,base_dim*4,base_dim*16,self.act_fn),
            BottleNeck(base_dim*16,base_dim*4,base_dim*16,self.act_fn),
            BottleNeck(base_dim*16,base_dim*4,base_dim*16,self.act_fn,down=True),
        )
        self.layer_5 = nn.Sequential(
            BottleNeck(base_dim*16,base_dim*8,base_dim*32,self.act_fn),
            BottleNeck(base_dim*32,base_dim*8,base_dim*32,self.act_fn),
            BottleNeck(base_dim*32,base_dim*8,base_dim*32,self.act_fn),
        )
        self.avgpool = nn.AvgPool2d(7,1)
        self.fc_layer = nn.Linear(base_dim*32,num_classes)
```

```
def forward(self, x):
    out = self.layer_1(x)
    out = self.layer_2(out)
    out = self.layer_3(out)
    out = self.layer_4(out)
    out = self.layer_5(out)
    out = self.avgpool(out)
    out = out.view(batch_size,-1)
    out = self.fc_layer(out)

    return out
```

Q&A

감사합니다.