합성곱신경망

https://sites.google.com/site/kyunghoonhan/deep-learning-ii

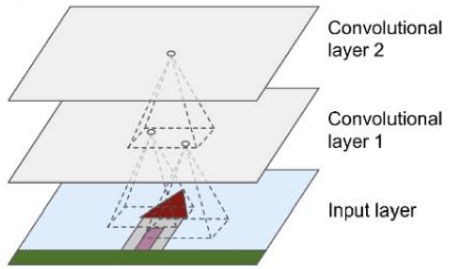
0. 합성곱신경망 구성

Conv2D(filters,kernel_size, activation...)
MaxPooling2D()
Conv2D
MaxPooling2D

..
.
.
Flatten()
Dense(
Dense(
Dense(최종output)

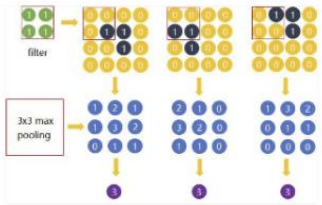
합성곱층

1. 필터와 편향을 학습시킨다.
2. 채널이 사라진다. 입력 데이터는 N×C×H×W 이고 출력데이터는 N×FN×OH×OW이다.
3. 필터를 훈련시켜 낮은 층의 필터는 저수준의 로컬한 특징을 찾아내고 높은 층의 필터는 더 고수준의 더 글로벌한 특징을 찾아내는 것이 목표다.



Max Pooling층

1. 학습시킬 parameter가 없다.
2. 채널별로 독립적으로 시행한다. 입력 데이터는 N×C×H×W이고 출력데이터는 N×C×OH×OW 이다.
3. DownSampling을 통하여 다음 합성곱층에서 더 빨리 글로벌한 특징을 찾아낼 수 있게 한다. 또한 파라미터 숫자를 줄여서 계산비용을 줄이고 overfitting을 억제한다. 또한 약간의 평행이동에 대하여 변하지 않도록 한다.



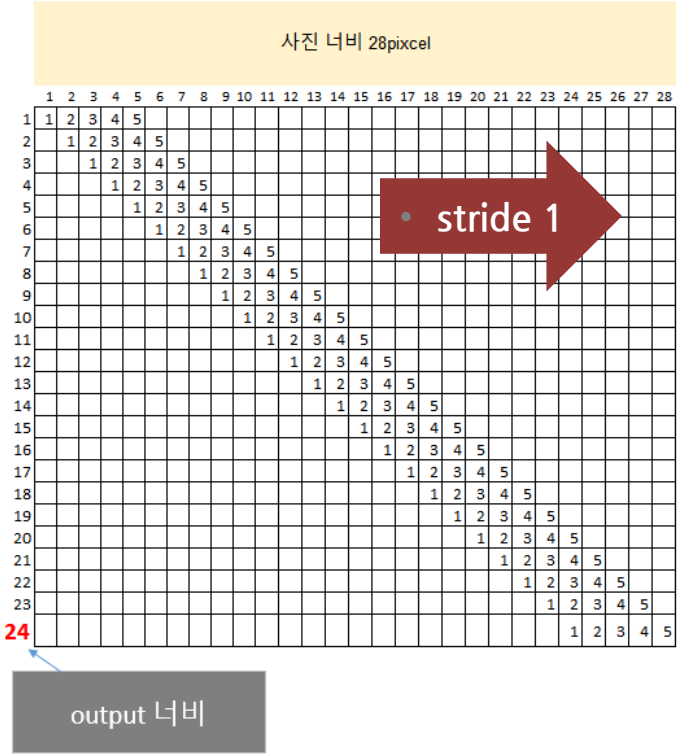
1. 합성곱신경망의 output shape 계산법

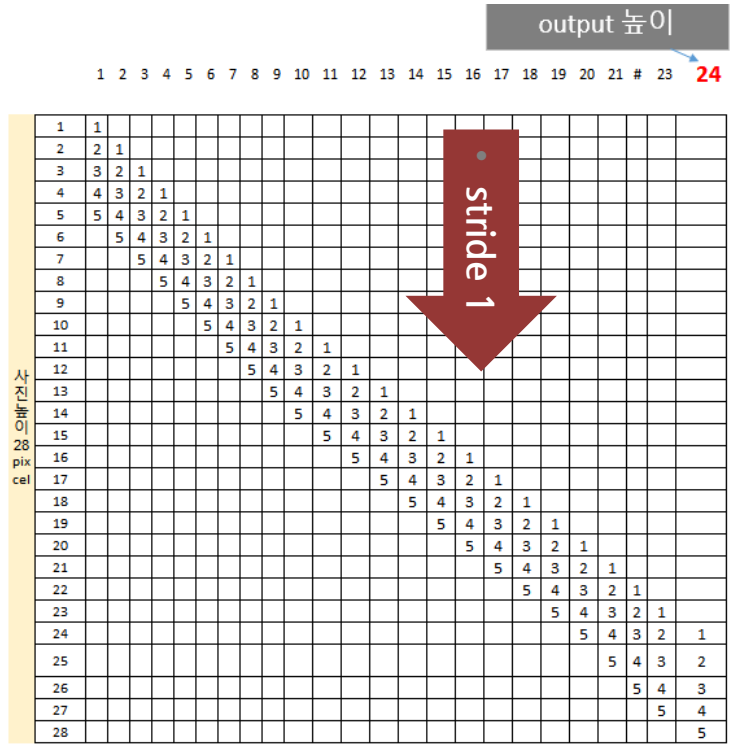
컨볼루션 레이어의 출력 텐서 크기

- 테스트를 실시했습니다.
  - O: 출력 이미지의 크기(너비)
  - I: 입력 이미지의 크기(너비)
  - K: Conv 레이어에서 사용하는 커널의 크기(너비)
  - N: 커널 수
  - S: 컨볼루션 연산의 스트라이드
  - P: 패딩 사이즈
- O(출력 이미지의 크기(너비))는 다음과 같다:

$$O = \frac{I - K + 2P}{S} + 1$$

- 아웃풋 이미지의 수는N)와 상담





output 사이즈 $\frac{28(\text{너비})-5+2*0}{1}+1$ $\frac{28(\text{높이})-5+2*0}{1}+1$

1. 합성곱신경망의 output shape 계산법

1. 손글씨 로드

- 가로, 세로 28,28의 1채널 흑백 이미지임 (합성곱신경망은 반드시 높이,너비,채널수 로 입력해야함)

```
1 import tensorflow as tf
2 from tensorflow.keras import datasets, layers, models
3 (train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()
4
5 train_images = train_images.reshape((60000, 28, 28, 1))
6 test_images = test_images.reshape((10000, 28, 28, 1))
7
8 train_images, test_images = train_images / 255.0, test_images / 255.0
```

2. 합성곱층 생성 filters=64, kernel_size=(5, 5) 로 작성해도 됨

```
1 model = models.Sequential()
2 model.add(layers.Conv2D(64, (5, 5), activation='relu', input_shape=(28, 28, 1)))
3 model.summary()
```

커널수(param계산에사용됨) 커널크기 높이,너비,채널수

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 24, 24, 64)        1664
=================================================================
Total params: 1,664
Trainable params: 1,664
Non-trainable params: 0
_________________________________________________________________
```

출력높이,출력너비,커널수

컨볼루션 레이어의 출력 텐서 크기

- 테스트를 실시했습니다.
  - O: 출력 이미지의 크기(너비)
  - I: 입력 이미지의 크기(너비) → 28
  - K: Conv 레이어에서 사용하는 커널의 크기(너비) → 5
  - N: 커널 수 → 64
  - S: 컨볼루션 연산의 스트라이드 → 1
  - P: 패딩 사이즈 → valid(0)
- O(출력 이미지의 크기(너비))는 다음과 같다:

$$O = \frac{I - K + 2P}{S} + 1$$

- 아웃풋 이미지의 수는N)와 상담

Output=(28-5+2*0) /1

최종=Output+1

24

1. 합성곱신경망의 output shape 계산법

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

커널크기

padding	one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input. When padding="same" and strides=1, the output has the same size as the input.

* 패딩에 대한 자세한 계산은 슬라이드 6~8p에서 진행함

패딩0 valid 생략

```
model = models.Sequential()
model.add(layers.Conv2D(64, (5, 5), activation='relu', input_shape=(28, 28, 1)))
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 24, 24, 64)        1664
=================================================================
Total params: 1,664
Trainable params: 1,664
Non-trainable params: 0
_________________________________________________________________
```

패딩있음. same

커널사이즈에 따라 다름

```
model = models.Sequential()
model.add(layers.Conv2D(64, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
model.summary()
```

커널크기에 따라 패딩 사이즈 달라짐

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 28, 28, 64)        1664
=================================================================
Total params: 1,664
Trainable params: 1,664
Non-trainable params: 0
_________________________________________________________________
```

1. 합성곱신경망의 output shape 계산법 (output size 계산)

문제1

사진 너비 28pixcel, 너비10, stride는 1, padding는 0 ==> (

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	2	3	4	5	6	7	8	9	10																		
	1	2	3	4	5	6	7	8	9	10																	
		1	2	3	4	5	6	7	8	9	10																
			1	2	3	4	5	6	7	8	9	10															
				1	2	3	4	5	6	7	8	9	10														
					1	2	3	4	5	6	7	8	9	10													
						1	2	3	4	5	6	7	8	9	10												
							1	2	3	4	5	6	7	8	9	10											
								1	2	3	4	5	6	7	8	9	10										
									1	2	3	4	5	6	7	8	9	10									
										1	2	3	4	5	6	7	8	9	10								
											1	2	3	4	5	6	7	8	9	10							
												1	2	3	4	5	6	7	8	9	10						
													1	2	3	4	5	6	7	8	9	10					
														1	2	3	4	5	6	7	8	9	10				
															1	2	3	4	5	6	7	8	9	10			
																1	2	3	4	5	6	7	8	9	10		
																	1	2	3	4	5	6	7	8	9	10	

문제2

사진 너비 28pixcel, 너비10, stride는 2, padding는 0 ==

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	2	3	4	5	6	7	8	9	10																		
		1	2	3	4	5	6	7	8	9	##																
				1	2	3	4	5	6	7	8	9	##														
						1	2	3	4	5	6	7	8	9	##												
								1	2	3	4	5	6	7	8	9	##										
										1	2	3	4	5	6	7	8	9	##								
												1	2	3	4	5	6	7	8	9	##						
														1	2	3	4	5	6	7	8	9	##				
																1	2	3	4	5	6	7	8	9	##		
																		1	2	3	4	5	6	7	8	9	10

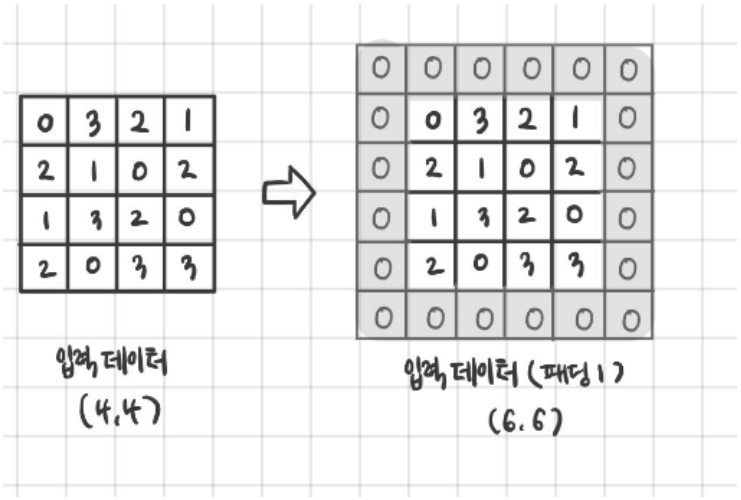
문제3

model.add(layers.Conv2D(64, 3, strides=(2), padding='same', activation='relu', input_shape=(100,100,1)))

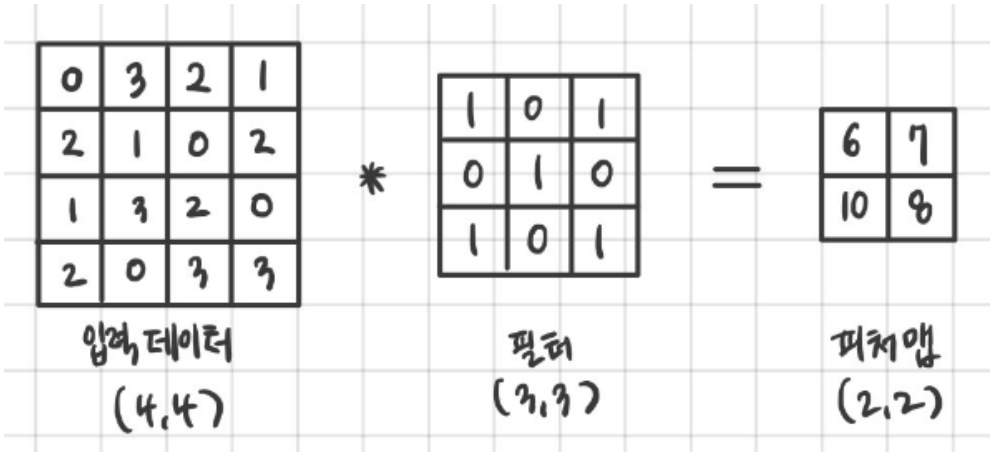
의 output_shape은? padding='same' 의 값은 1임 (패딩은 커널크기에 따라 결과가 달라짐) 여기에서는 1로 계산됨

1. 합성곱신경망의 output shape 계산법 (output size 계산) - padding

Padding 출력 크기를 보정하기 위해 사용하며 '충전재'라는 의미처럼 입력 데이터의 사방을 특정 값으로 채우는 것을 말한다. 일반적으로 특정 값은 그냥 0을 사용하기에 제로패딩(zero padding)이라고도 한다. 아래의 사진은 (4,4)인 입력데이터에 폭이 1 pixel인 패딩을 적용하여 (6,6) 크기가 된 그림이다.



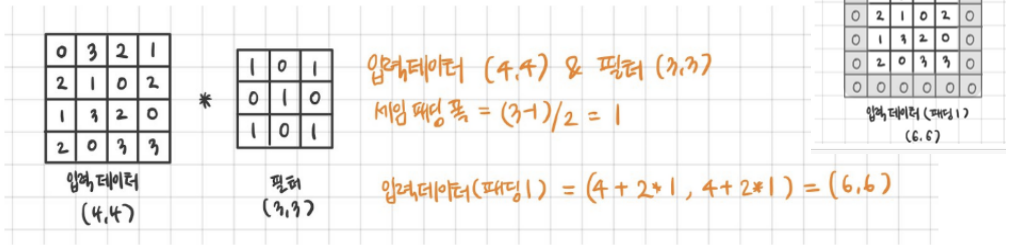
padding='valid'



1. 합성곱신경망의 output shape 계산법 (output size 계산)

same Padding

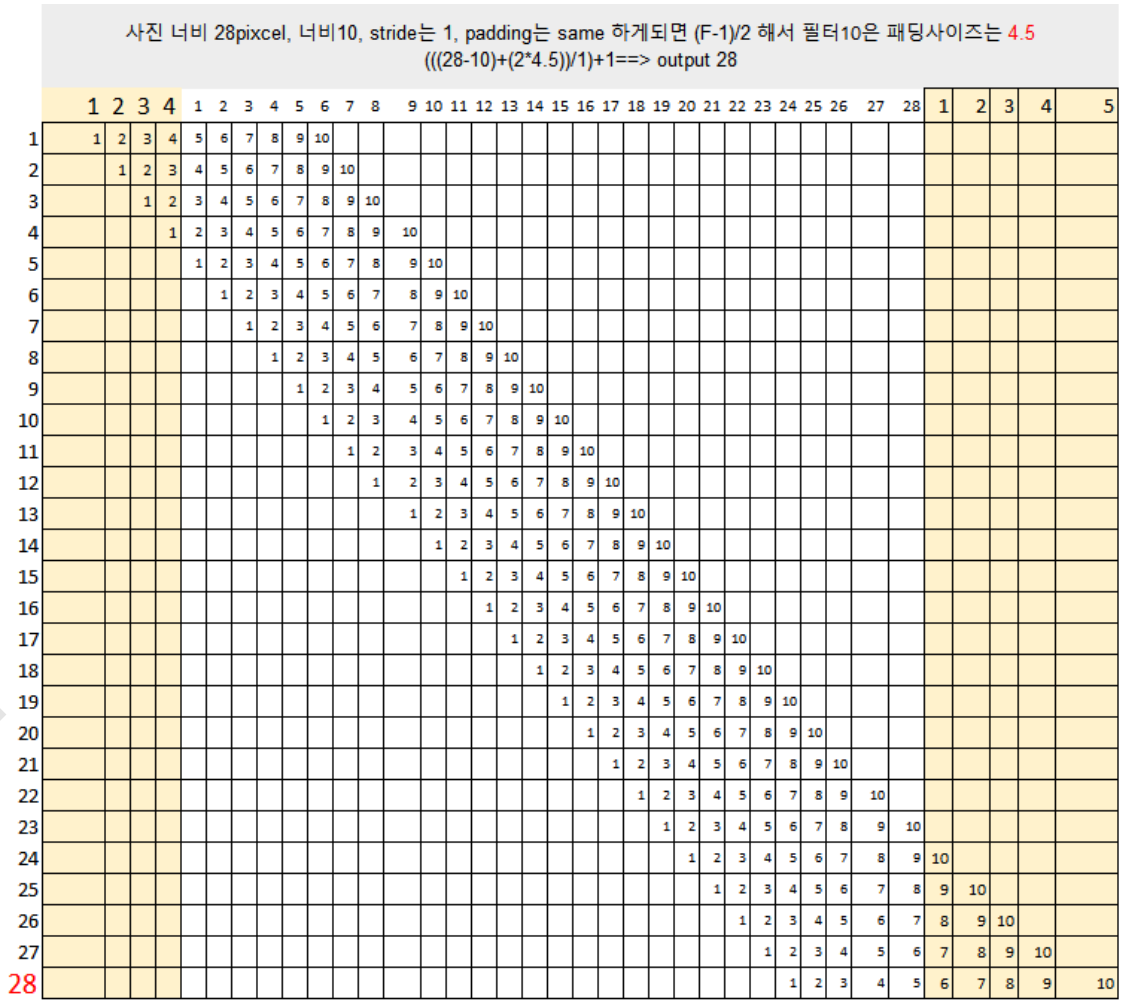
세임 패딩은 출력 크기를 입력 크기와 동일하게 유지한다. 입력데이터 (Hi,Wi) 와 (F, F) 사이즈의 필터가 있을때 세임 패딩의 폭 P는 (F-1)/2가 된다. 그러므로 세임 패딩을 적용한 입력 데이터의 크기는 (Hi+2*P, Wi+2*P) 가 된다. 이처럼 세임 패딩은 풀패딩의 절반 개념이므로 절반 패딩(half padding) 라고도 부른다. 마찬가지로 위의 예시를 이용해 값을 계산해보면 아래와 같다.




```
model = models.Sequential()
model.add(layers.Conv2D(64, kernel_size=(10, 10), padding='same',
                        activation='relu', input_shape=(28, 28, 1)))
model.summary()
```

```
Model: "sequential_30"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_71 (Conv2D)           (None, 28, 28, 64)        6464

=================================================================
Total params: 6,464
Trainable params: 6,464
Non-trainable params: 0
_________________________________________________________________
```

사진 너비 28pixcel, 너비10, stride는 1, padding는 same 하게되면 (F-1)/2 해서 필터10은 패딩사이즈는 4.5
(((28-10)+(2*4.5))/1)+1==> output 28

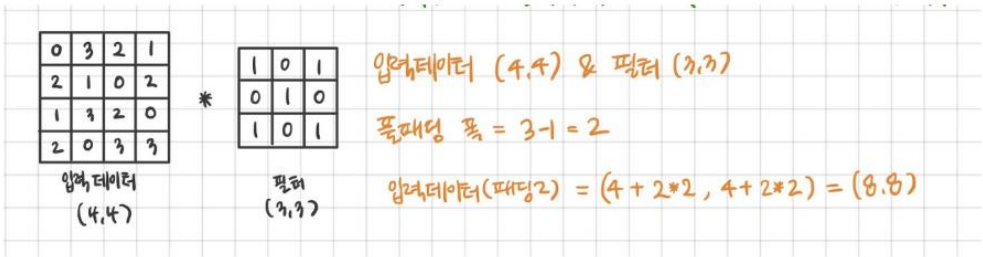


1. 합성곱신경망의 output shape 계산법 (output size 계산)

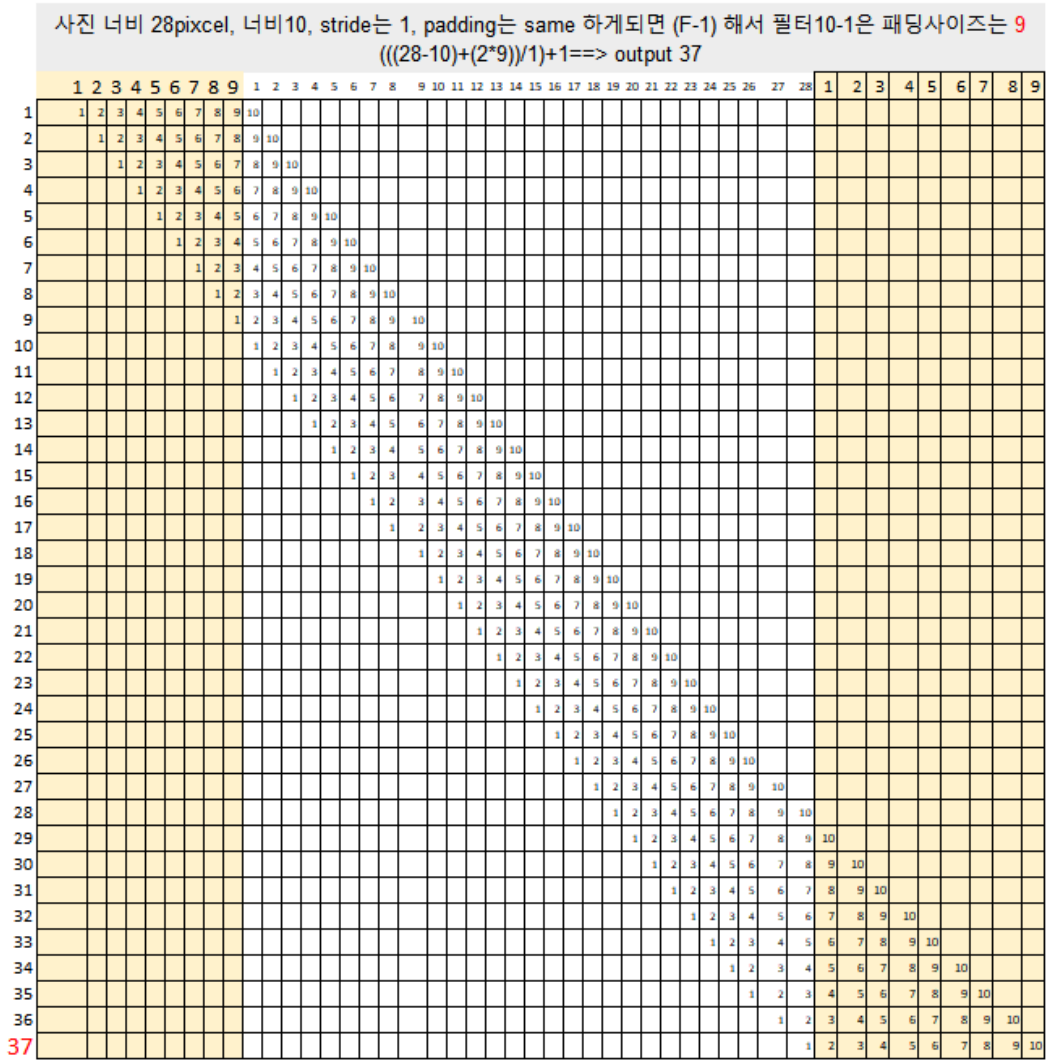
full Padding

먼저 밸리드 패딩은 실질적으로 입력데이터 주위를 0으로 채우지 않아 언급한 문제들이 발생하기에 잘 사용하지 않는다. 입력데이터의 모든 원소가 합성곱 연산에 같은 비율로 참여하도록 하는 패딩 방식을 풀 패딩(full padding)이라고 한다. 위에서 등장한 (4,4)의 입력데이터와 (3,3)의 필터가 주어진 경우 풀패딩을 적용한 입력 데이터의 모습은 아래와 같다.

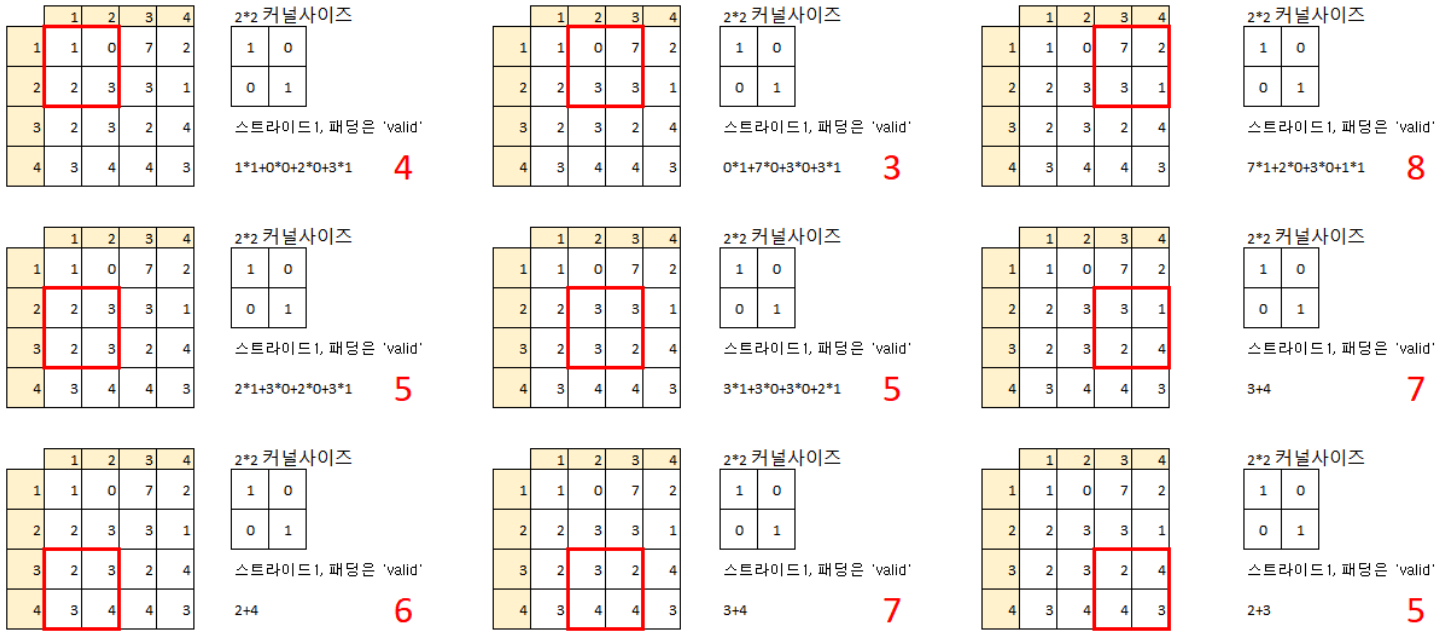
풀패딩에서 패딩의 폭이 2 pixel로 도출된 과정을 설명해 보자면 입력데이터 (Hi,Wi) 와 필터 (F, F)가 있을때 풀 패딩의 폭 P는 F-1이 된다. 그러므로 풀패딩을 적용한 입력데이터의 크기는 (Hi+2*P, Wi+2*P)가 되는 것이다. 예시를 통해 실제 값을 계산해보자.



https://stackoverflow.com/questions/37674306/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max-pool-of-t



1. 합성곱신경망의 output shape 계산법 (output size 계산)



((4-2)+(2*0) /1) + 1 => output3

4	3	8
5	5	7
6	7	5

https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D

MaxPool2D

```
tf.keras.layers.MaxPool2D(
    pool_size=(2, 2),
    strides=None,
    padding='valid',
    data_format=None,
    **kwargs
)
```

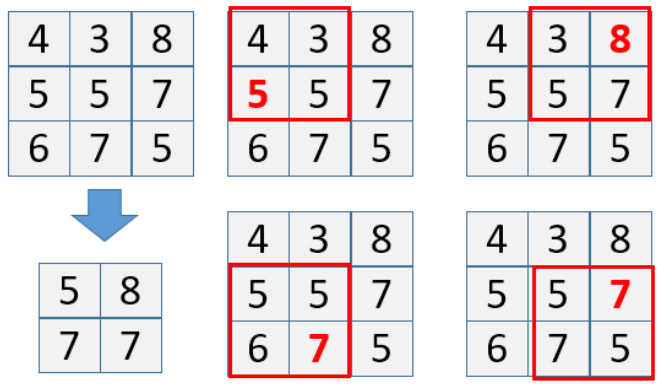
strides는 커널사이즈에
따라 달라짐 기본

maxool2d도 합성곱 outpt 계산식 적용함

((input - 커널) + (2*패딩) / 스트라이드)

+1

maxPool(2,2), stride=1



1. 합성곱신경망의 output shape 계산법 (output size 계산)

MaxPooling2D 기본은 (2,2) , 스트라이드 2 임 ((input - 2) *(2*0) / 2)+1

MaxPooling2D(2,2), strides=2

```
model = models.Sequential()
model.add(layers.Conv2D(64, kernel_size=(2, 2),
                        activation='relu', input_shape=(4, 4, 1
model.add(layers.MaxPooling2D((2, 2),strides=2))
model.summary()
```

```
Model: "sequential_40"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_83 (Conv2D)           (None, 3, 3, 64)          320

max_pooling2d_57 (MaxPoolin  (None, 1, 1, 64)          0
g2D)

=================================================================
Total params: 320
Trainable params: 320
Non-trainable params: 0
_________________________________________________________________
```

MaxPooling2D(2,2), strides=1

```
model = models.Sequential()
model.add(layers.Conv2D(64, kernel_size=(2, 2),
                        activation='relu', input_shape=(4, 4, 1)))
model.add(layers.MaxPooling2D((2, 2),strides=1))
model.summary()
```

```
Model: "sequential_39"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_82 (Conv2D)           (None, 3, 3, 64)          320

max_pooling2d_56 (MaxPoolin  (None, 2, 2, 64)          0
g2D)

=================================================================
Total params: 320
Trainable params: 320
Non-trainable params: 0
_________________________________________________________________
```

MaxPooling2D(2,2), strides=2,padding='same'

```
model = models.Sequential()
model.add(layers.Conv2D(64, kernel_size=(2, 2),
                        activation='relu', input_shape=(4, 4, 1)))
model.add(layers.MaxPooling2D((2, 2),strides=2,padding='same'))
model.summary()
```

```
Model: "sequential_41"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_84 (Conv2D)           (None, 3, 3, 64)          320

max_pooling2d_58 (MaxPoolin  (None, 2, 2, 64)          0
g2D)

=================================================================
Total params: 320
Trainable params: 320
Non-trainable params: 0
_________________________________________________________________
```

평균 풀링보다 최대 풀링을 거 선호한다. 이유는 평균 풀링은 합성곱층을 통과하는 특징들을 희석시킬 가능성이 높기 때문이다. 즉 입력에서 합성곱 필터가 특성의 값을 상쇄시키기 때문이다. 최대 풀링은 가장 큰 특징을 유지시키는 성질이 있음으로 이미지 분류 작업에 잘 맞는다.

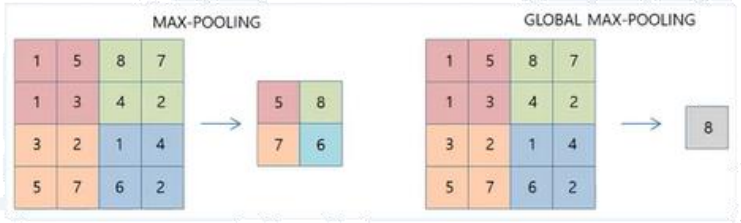
1. 참고: pooling

General pooling layer

→ Filter size와 stride를 갖는다.
→ 필터 사이즈 내에 down sampling 가능한 aggregation함수를 사용한다. (min, max, average 등).

Global pooling layer

→ Filter size와 stride를 갖지 않고, 모든 범위에 대하여 pooling을 한다.
→ 전체에 down sampling 가능한 aggregation함수를 사용한다. (min, max, average 등)



https://blog.naver.com/PostView.naver?blogId=qbxlvnf11&logNo=221932118708&parentCategoryNo=&categoryNo=74&viewDate=&isShowPopularPosts=false&from=postView

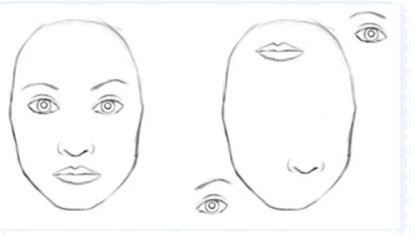
최초의 CNN은 pooling layer를 기본으로 가지고 있지만,

최근 연구 트랜드는 다른 cost를 줄일 수 있는 방법이 나오거나 컴퓨팅 파워가 더 나아져,
과감히 pooling을 쓰지 않기도 한다!

한편, pooling layer가 다운샘플링을 함으로서 field of view (시야각)을 넓히는 역할을 한다. (넓은 범위를 볼 수 있다.)

→ 그런데 생기는 문제점?
CNN은 오른쪽의 이미지들을 같은 얼굴이라고 인식한다.

(실제 사람의 시각 프로세스는 CNN과 다르게 그렇지 않다..)



→ 이를 해결하기 위해, capsule net 등 아이디어가 나오고 있다.

한편, data augmentation (cut mix 등)기법으로 어느정도 해결 가능 하다.

https://blog.naver.com/PostView.naver?blogId=qbxlvnf11&logNo=222105994766&parentCategoryNo=&categoryNo=74&viewDate=&isShowPopularPosts=false&from=postView

1. 합성곱신경망의 output shape 계산법 (output size 계산)

[1], [2]의 output 계산

```
1  model = models.Sequential()
2  model.add(layers.Conv2D(64, kernel_size=(2, 2),
3                          activation='relu', input_shape=(30, 30, 1)))
4  model.add(layers.MaxPooling2D())
5  model.summary()
6
```

1

2

1. 합성곱신경망의 output shape 계산법 (계산하여 보세요)

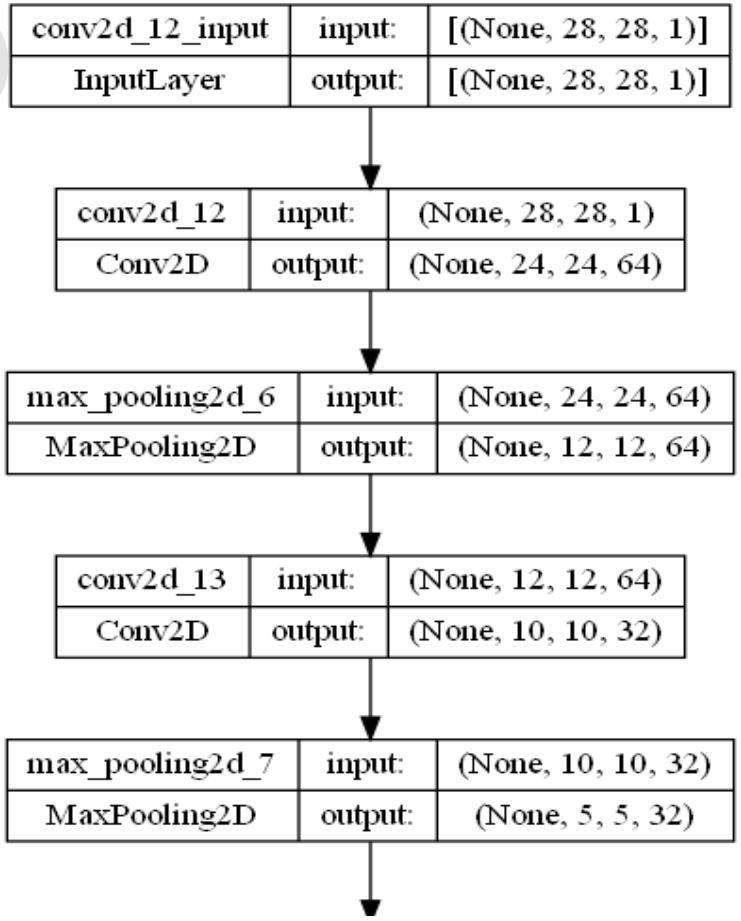
*아래의 커널크기(filters)는 output 공부를 위하여 다양하게 넣은 의미 없는 숫자임

```
model = models.Sequential()
model.add(layers.Conv2D(64, (5, 5), activation='relu',
          input_shape=(28, 28, 1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(32, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(120, (3, 3), activation='relu'))

model.add(layers.Flatten())

model.add(layers.Dense(12, activation='relu')

model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

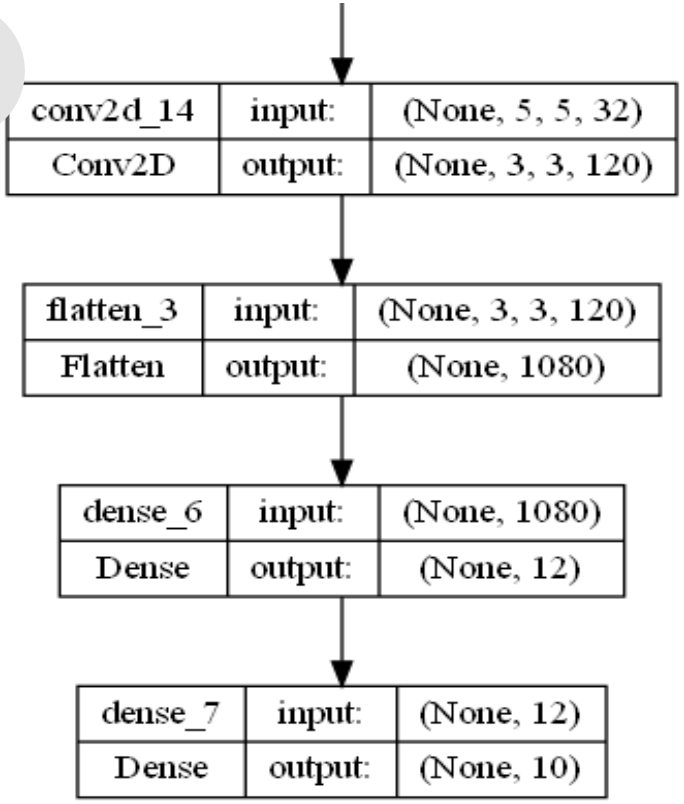
1. 합성곱신경망의 output shape 계산법

```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model.png')
plot_model(model, to_file='model_shapes.png', show_shapes=True)
```

1



2



2. 합성곱신경망의 Param 계산

Convolution layer의 parameter 갯수

- CNN의 각 layer는 weight parameter와 bias parameter가 존재.
- 전체 네트워크의 parameter 수는 각 conv layer 파라미터 수의 합
- 각각 기호를 아래와 같이 정의
  - W_c: Number of weights of the Conv layer
  - B_c: Number of biases of the Conv layer
  - P_c: Number of parameters of the Conv layer
  - K: Size(width) of kernels used in the Conv layer
  - N: Number of kernels
  - C: Number of channels of the input image

$$W_c = K^2 \times C \times N$$
$$B_c = N$$
$$P_c = W_c + B_c$$

- Conv layer에서 모든 커널의 깊이는 항상 입력 이미지의 채널 수와 같음
- 따라서 모든 커널에는 $K^2 \times C$개의 parameter들이 있으며, 그러한 커널들이 N개 존재

Example on AlexNet

- AlexNet의 Conv-1에 대해
  - 입력 이미지의 채널 수 $C = 3$
  - Kernel size $K = 11$
  - 전체 커널 개수 $N = 96$
- 따라서 파라미터의 갯수는 아래와 같이 정의됨

$$W_c = 11^2 \times 3 \times 96 = 34,848$$
$$B_c = 96$$
$$P_c = 34,848 + 96 = 34,944$$

- Conv-2/3/4/5도 동일한 방법으로 각각 614,656/885,120/1,327,488/884,992개의 parameter를 갖는것을 계산 가능
- AlexNet conv layer의 parameter 개수는 3,747,200개
- FC layer의 parameter 수가 더해지지 않았으므로 전체 네트워크의 parameter 개수가 아님
- Conv layer의 장점은 weight parameter가 공유되므로 FC layer에 비해 매개변수가 훨씬 작다는 장점이 있음

```
1  model = models.Sequential()
2  model.add(layers.Conv2D(64, (5, 5), activation='relu',
3                          input_shape=(28, 28, 1)))
4  model.add(layers.MaxPooling2D())
5  model.add(layers.Conv2D(32, (3, 3), activation='relu'))
6  model.add(layers.MaxPooling2D((2, 2)))
7  model.add(layers.Conv2D(120, (3, 3), activation='relu'))
8  model.add(layers.Flatten())
9  model.add(layers.Dense(12, activation='relu'))
10 model.add(layers.Dense(10, activation='softmax'))
11 model.summary()
```

커널(필터)	input_size	(커널size)	채널수	Wc 커널사이즈의제곱*채널갯수*커널수		Bc 커널수	param (Wc+Bc)
64	28	5	1	(10**2)*1*32	1600	64	1664
32		3	64	(3**2)*32*64	18432	32	18464
120		3	32	36864	34560	120	34680

3. 미션 output Shape와 param 계산

```
model = models.Sequential()
from tensorflow.keras.layers import Dense, Input, Conv2D, Conv2DTranspose, Flatten, Reshape
from tensorflow.keras.models import Model


inputs=Input(shape=(28,28,3))
x=Conv2D(32,3,activation='relu',padding='same')(inputs)
x=Conv2D(64,3,activation='relu',padding='same')(x)
x=Flatten()(x)
last=Dense(10)(x)
x=Dense(28*28*64)(last)
```

수식	input_size	output _shape	Padding (same) (커널size-1)/2	커널(필터)	(커널size)	채널수	Wc 커널사이즈의제곱*채널갯수*커널수		Bc 커널수	param (Wc+Bc)
Conv2D(32,3,padding='same')										
Conv2D(64,3, padding='same')										
Flatten()										
Dense(10)(x)										

3. 미션 output Shape와 param 계산

```
inputs=Input(shape=(28,28,3))
x=Conv2D(32,3,activation='relu',padding='same')(inputs)
x=Conv2D(64,3,activation='relu',padding='same')(x)
x=Flatten()(x)
last=Dense(10)(x)

x=Dense(28*28*64)(last)
```

100,75 에서 높이 100 계산

수식	input_size	output _shape	Padding (same) (커널size-1)/2	커널(필터)	(커널size)	채널수	Wc 커널사이즈의제곱*채널갯수*커널수		Bc 커널수	param (Wc+Bc)
Conv2D(32,3,2,padding='same')										
Conv2D(64,3,2, padding='same')										
Conv2D(128,3,2, padding='same')										

100,75 에서 너비75 계산

수식	input_size	output _shape	Padding (same) (커널size-1)/2	커널(필터)	(커널size)	채널수	Wc 커널사이즈의제곱*채널갯수*커널수		Bc 커널수	param (Wc+Bc)
Conv2D(32,3,2,padding='same')										0
Conv2D(64,3,2, padding='same')										0
Conv2D(128,3,2, padding='same')										0

최종 output	
input_shape=(100,75,3)	
Conv2D(32,3,2,padding='same')	
Conv2D(64,3,2, padding='same')	
Conv2D(128,3,2, padding='same')	

flatten() => 13*10*128	
Dense(10) => (16641*10)	

4. 참고: 오토인코더를 위한 모델 구성

```
model = models.Sequential()
from tensorflow.keras.layers import Dense, Input, Conv2D, UpSampling2D, Flatten, Reshape
from tensorflow.keras.models import Model


inputs=Input(shape=(100,75,3))
x=Conv2D(32,3,2,activation='relu',padding='same')(inputs)
x=Conv2D(64,3,2, activation='relu',padding='same')(x)
x=Conv2D(128,3,2, activation='relu',padding='same')(x)
x=Flatten()(x)
latent=Dense(10)(x)
x=Dense(28*28*64)(last)
x = Dense((13 * 10 * 128))(latent)
x = Reshape((13, 10, 128))(x)
x = UpSampling2D(size = (2,2))(x)
x = Conv2D(128, (2,2), (1,1), activation='relu', padding='valid')(x)
x = UpSampling2D(size = (2,2))(x)
x = Conv2D(64, (1,1), (1,1), activation='relu', padding='valid')(x)
x = UpSampling2D(size = (2,2))(x)
x = Conv2D(32, (1,2), (1,1), activation='relu', padding='valid')(x)
x = Conv2D(1, (1,1), (1,1), activation='sigmoid')(x)

model=Model(inputs,x)
model.summary()
```

```
Model: "model_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_16 (InputLayer)        [(None, 100, 75, 3)]      0

conv2d_81 (Conv2D)           (None, 50, 38, 32)        896

conv2d_82 (Conv2D)           (None, 25, 19, 64)        18496

conv2d_83 (Conv2D)           (None, 13, 10, 128)       73856

flatten_25 (Flatten)         (None, 16640)             0

dense_50 (Dense)             (None, 10)                166410

dense_52 (Dense)             (None, 16640)             183040

reshape_6 (Reshape)          (None, 13, 10, 128)       0

up_sampling2d_1 (UpSampling  (None, 26, 20, 128)       0
2D)

conv2d_84 (Conv2D)           (None, 25, 19, 128)       65664

up_sampling2d_2 (UpSampling  (None, 50, 38, 128)       0
2D)

conv2d_85 (Conv2D)           (None, 50, 38, 64)        8256

up_sampling2d_3 (UpSampling  (None, 100, 76, 64)       0
2D)

conv2d_86 (Conv2D)           (None, 100, 75, 32)       4128

conv2d_87 (Conv2D)           (None, 100, 75, 1)        33
```

5. XAI(eXplainable AI, 설명가능 인공지능)

XAI는 사람이 AI의 동작과 최종결과를 이해하고 올바르게 해석할 수 있고, 결과물이 생성되는 과정을 설명 가능하도록 해주는 기술을 의미한다

▷ XAI를 둘러싼 다양한 움직임

•인공지능이 중요작업(mission critical)에 사용될 경우 인공지능의 설명성, 투명성 확보 기술, 기준 정립이 필요하다. 2018년 5월 28일, 의사 결정 이유에 대한 설명을 요구하는 EU의 일반 개인정보보호법(General Data Protection Regulation)이 발효되어 의사 결정 이유를 설명할 수 없는 AI 기술은 향후 의료, 군사 등 중요작업에는 사용하기 어렵게 될 것으로 예상된다.

•XAI는 인터넷의 핵심 기술을 개발한 미국 국방성의 연구개발부서 DARPA(Defense Advanced Research Projects Agency)가 주도하고 있다. 과기정통부도 XAI의 중요성을 인식하고 'I-Korea 4.0 실현을 위한 인공지능 R&D 전략(2018.5)'의 R&D 로드맵에 2025년까지 설명가능 학습, 추론 기술 개발 추진 계획을 포함했다.

https://blog.naver.com/PostView.naver?blogId=laonple&logNo=222389644140

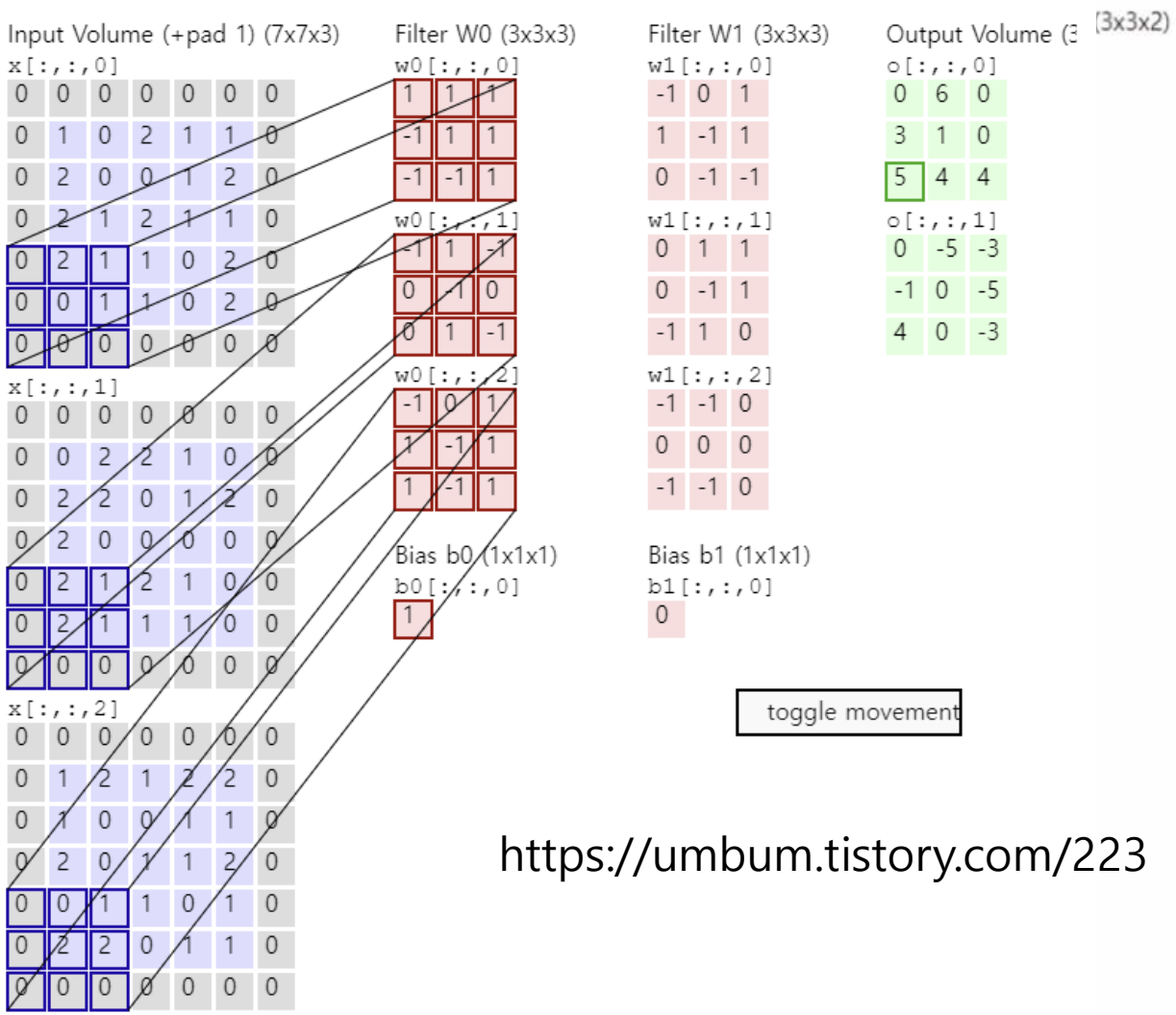
https://realblack0.github.io/2020/04/27/explainable-ai.html

http://www.sbr.ai/news/articleView.html?idxno=1632

http://www.aitimes.com/news/articleView.html?idxno=136219

https://github.com/sicara/tf-explain

참고: CNN 합성곱 신경망 output 값



https://umbum.tistory.com/223

```
Conv2D(2,
       3,
       1,
       padding='same')
```

참고: CNN 합성곱 신경망 output 값

``

채널값* 값(빨간색테두리)의 합 + Bias

Bias	1	0	6	0	3	1	0

R채널 커널(3,3)

1	1	1
-1	1	1
-1	-1	1

-1

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

4

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

-3

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

4

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

1

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

1

0	0	0	0	0	0	0
0	1	0	2	1	1	0
0	2	0	0	1	2	0
0	2	1	2	1	1	0
0	2	1	1	0	2	0
0	0	1	1	0	2	0
0	0	0	0	0	0	0

G채널 커널(3,3)

-1	1	-1
0	-1	0
0	1	-1

0

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

-3

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

2

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

-1

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

-2

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

1

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	2	0	1	2	0
0	2	0	0	0	0	0
0	2	1	2	1	0	0
0	2	1	1	1	0	0
0	0	0	0	0	0	0

B채널 커널(3,3)

-1	0	1
1	-1	1
1	-1	1

0

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

4

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

0

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

-1

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

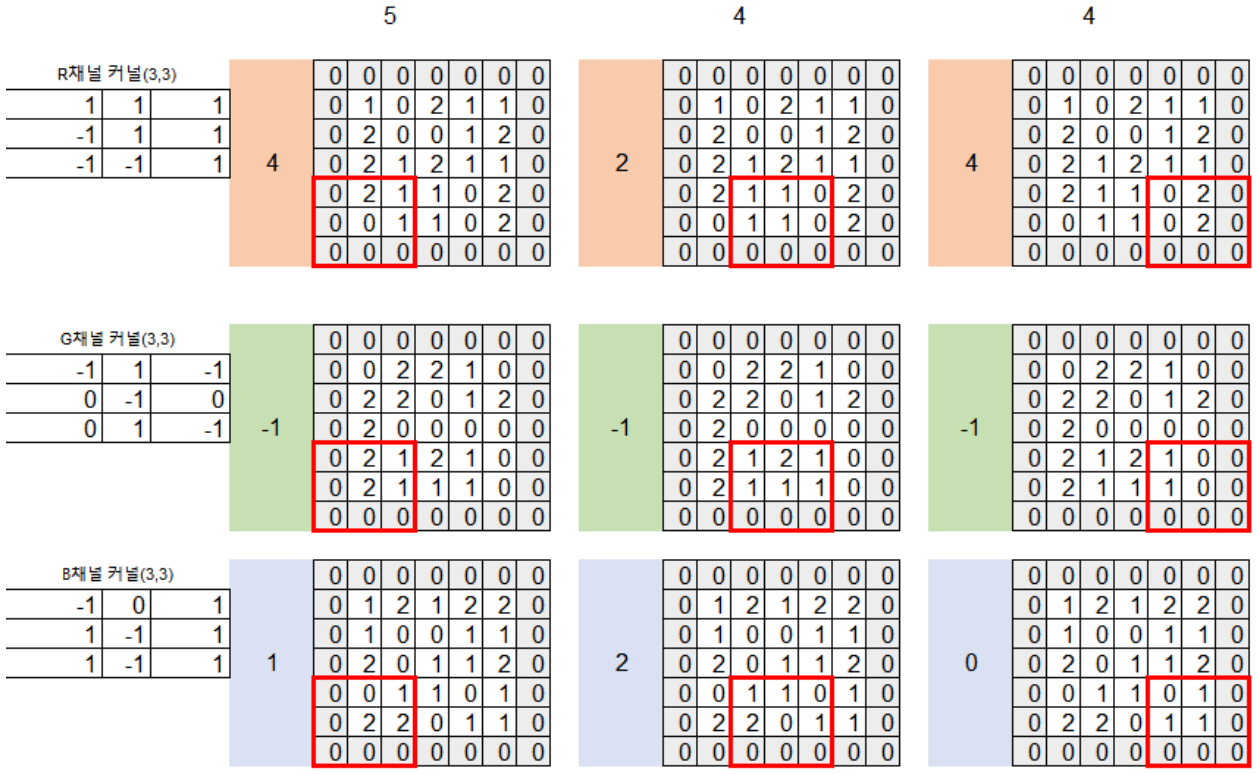
1

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

-3

0	0	0	0	0	0	0
0	1	2	1	2	2	0
0	1	0	0	1	1	0
0	2	0	1	1	2	0
0	0	1	1	0	1	0
0	2	2	0	1	1	0
0	0	0	0	0	0	0

참고: CNN 합성곱 신경망 output 값



0	6	0
3	1	0
5	4	4

output

input (5), stride(2), padd=1

kernel_size=3

⇒ (5-3)+(2*1) /2

⇒ 2+1=3

param

Wc=3**2=>9 * 3(채널수) * 1=27

Bc=1

Pc=27+1 => 28

5. 설명가능한 CNN 기본 (블랙박스 설명하기)

```
print(np.shape(feature_maps))
# 채널이 32개 있음
plt.figure(figsize=(15,3))
cnt=0
for x in range(32):
    plt.subplot(2,16,cnt+1)
    plt.imshow(feature_maps[0][:,:,x],cmap='gray')
    cnt+=1
plt.show()
```

```
(1, 26, 26, 32)
```