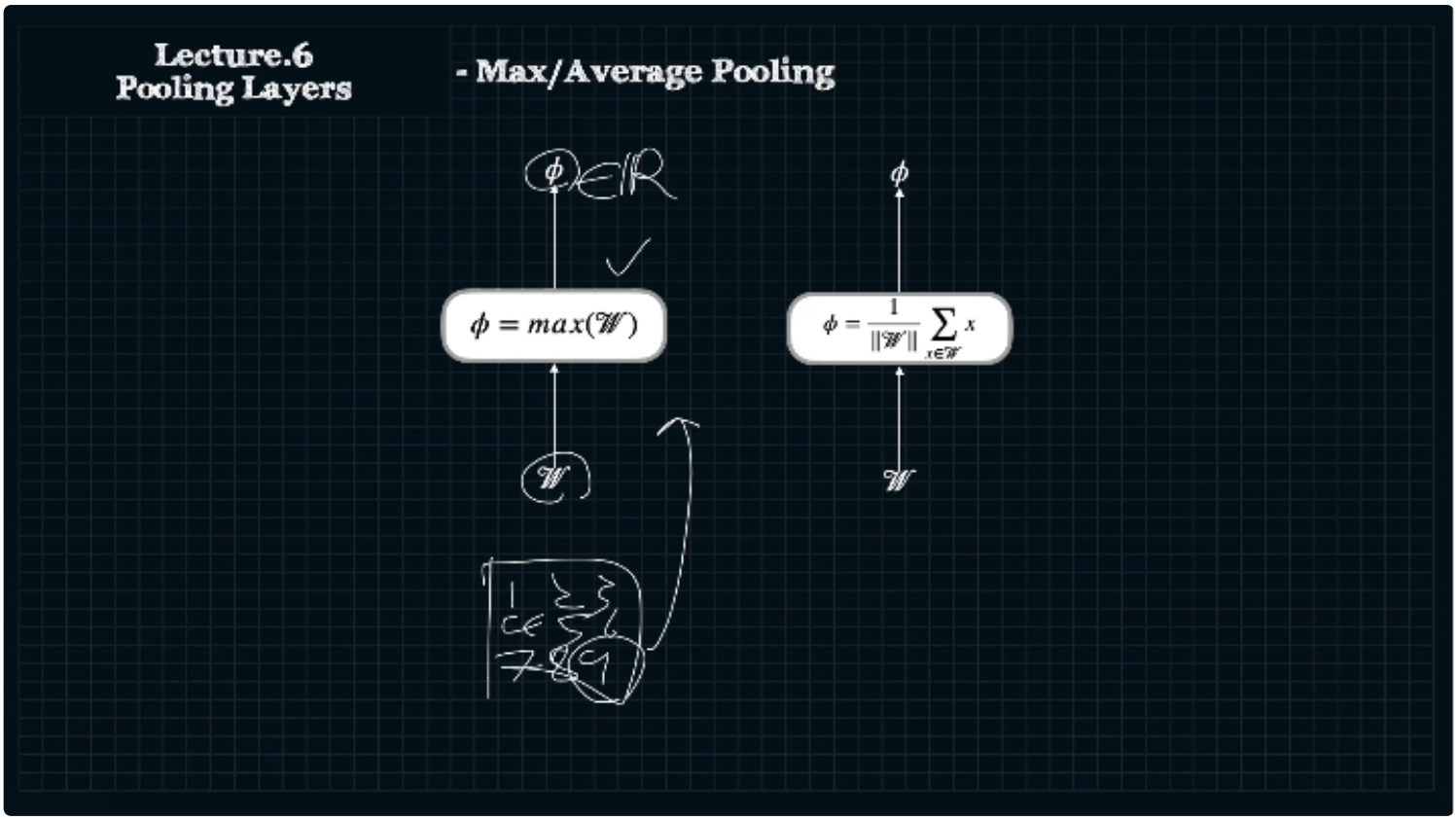
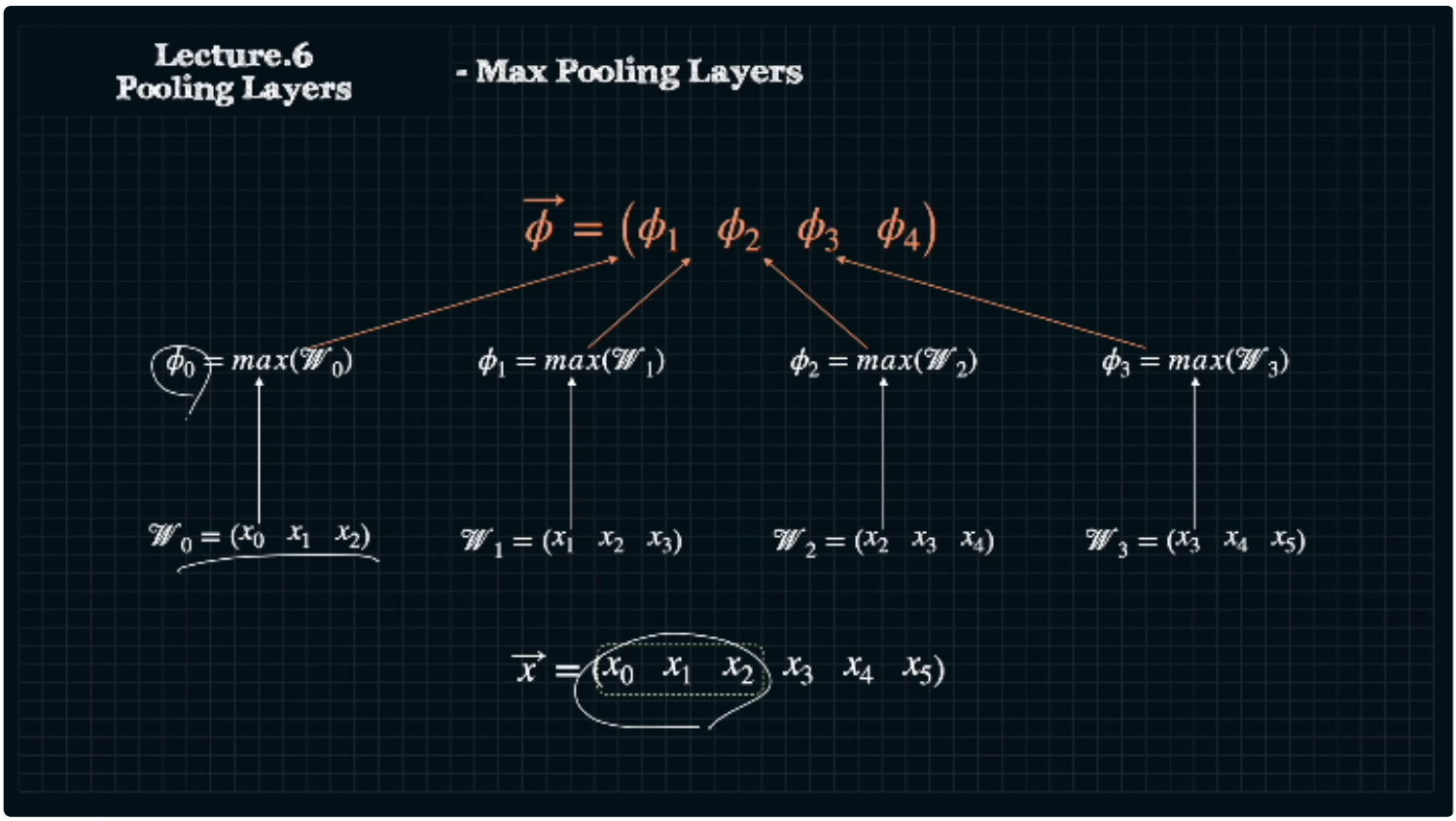
이미지 내에서 window가 지나다니면서 최대값을 뽑는것이다.



1, 2, 3, 4, 5 ,6,7,8, 9를 Window가 뽑았다면, 9가 뽑히는 것이다.

앞에서 다뤘던 Conv와 공통점은 Window를 뽑는다. scalar값을 만들어 낸다. 오른쪽은 max pooling인데 window에서 최대값 뽑고 나머지 다 더해서 최대 값으로 나눠주는것이다.

차이점은 Convolution layer같은경우에는 Convolution연산과 activation function을 통과하는 반면에 max pooling은 그냥 최대값 뽑고 평균을 만든다.

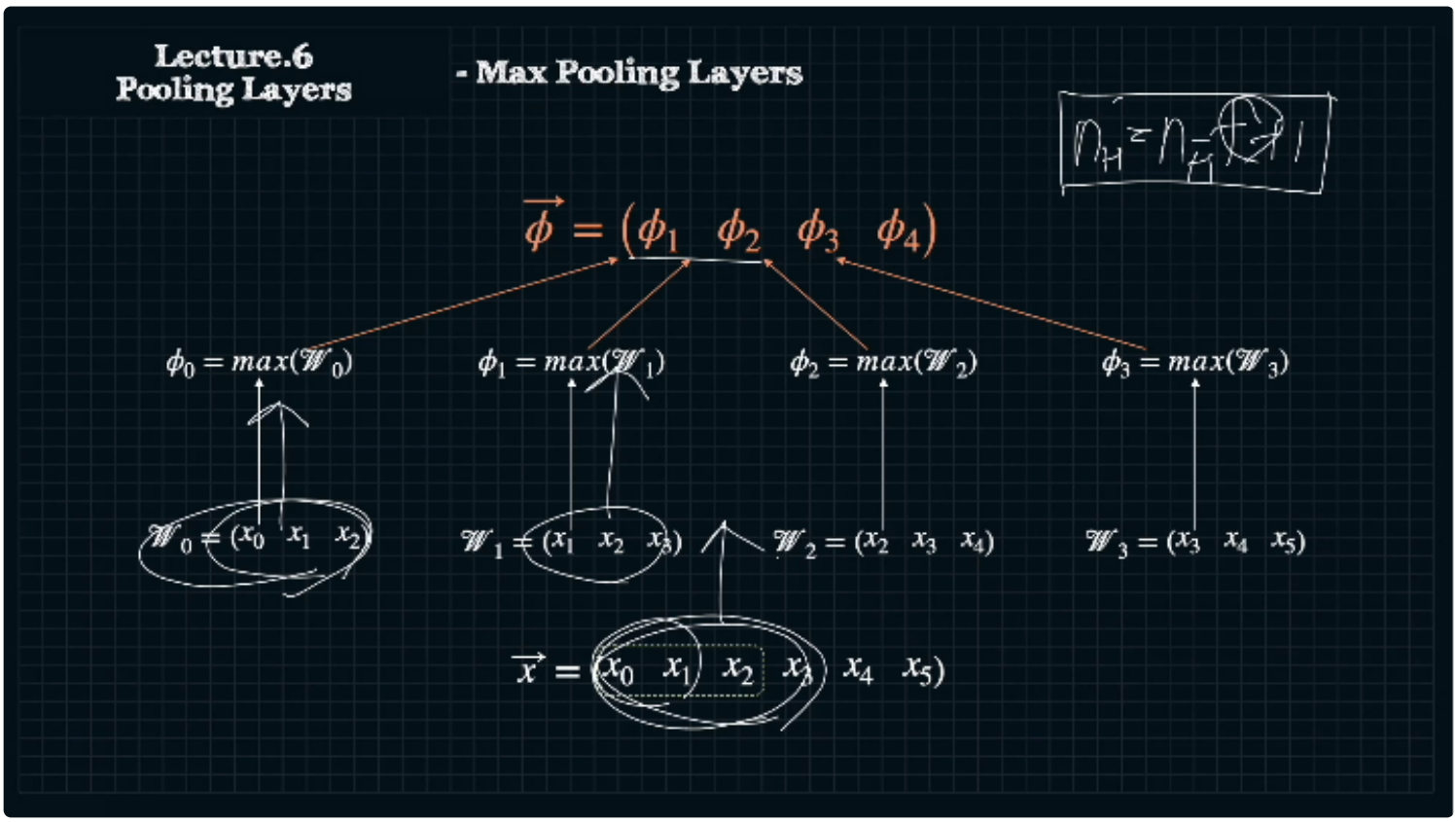


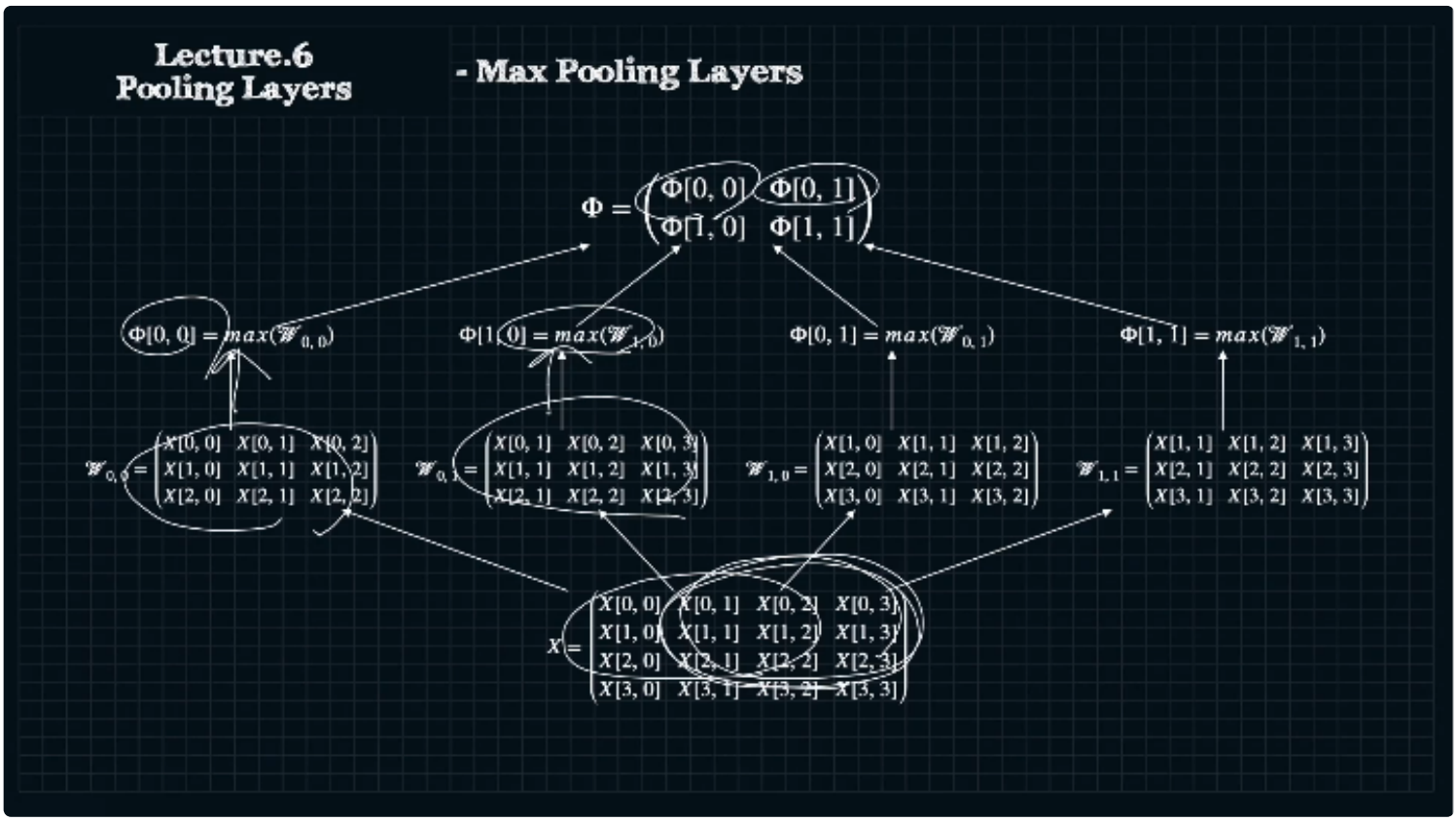
일차원에서 Max pooling layer가 작동하는 방법

첫 번째에서 최대값 뽑고 두 번째 window에서 최대값뽑고 마지막 N 번째까지 최대값 뽑는다.

이것을 다모아둔게 주황색이다.

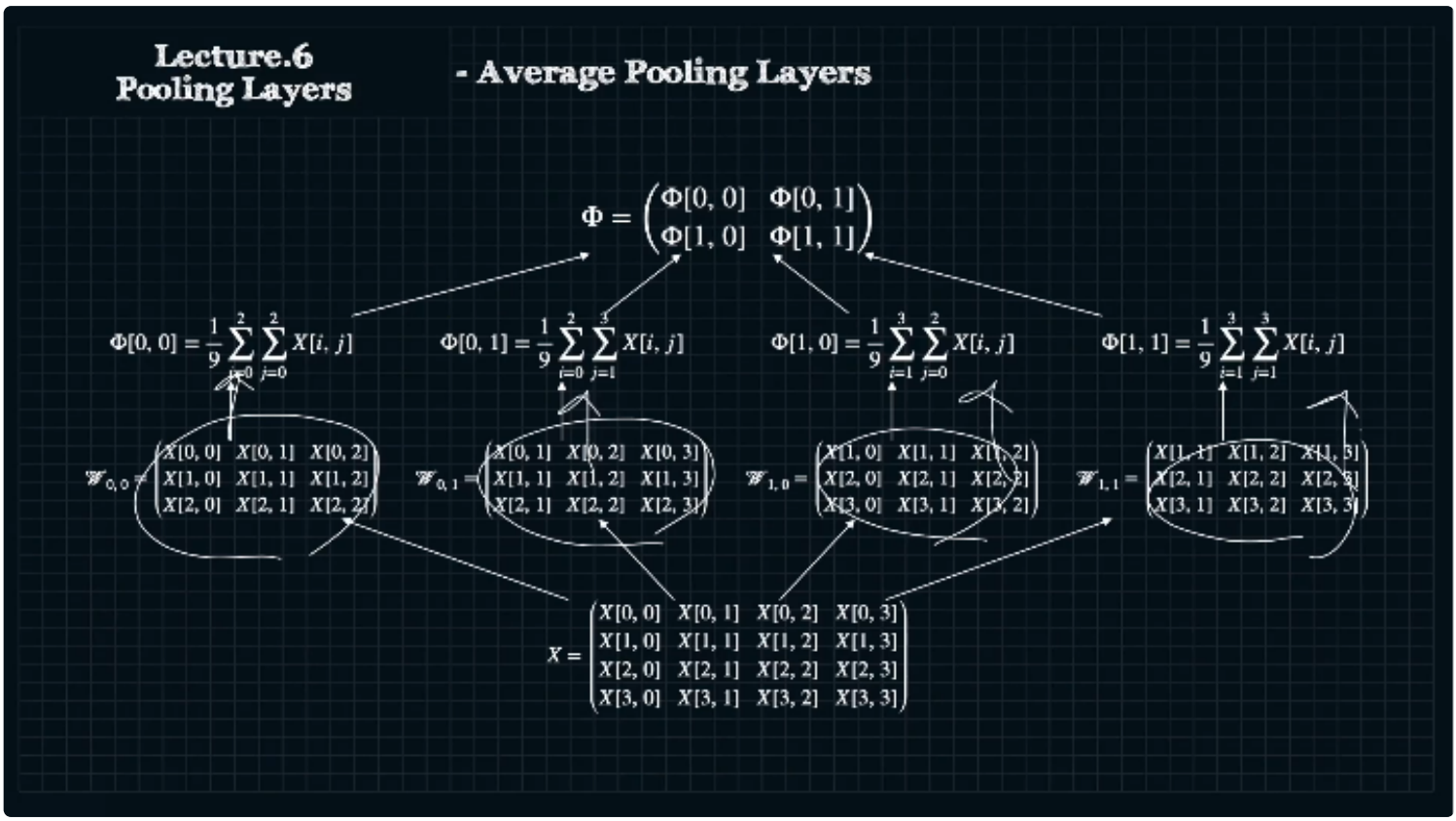
max pooling에서는 kernel bias를 통과시키지 않는다. 하지만, 공통점은 window 뽑고 scalar를 만드니까 shape를 만드는 방법은 동일하다.

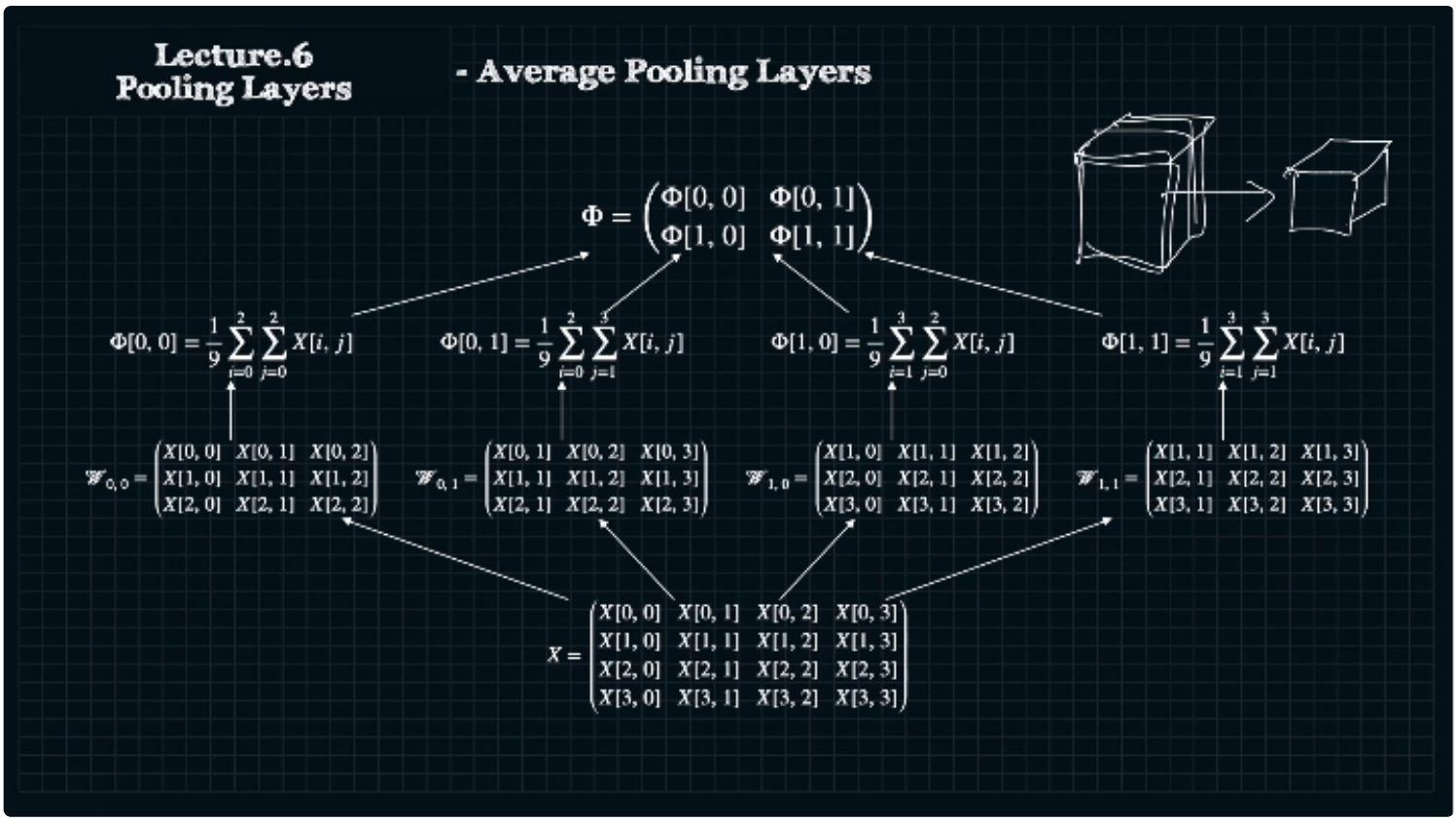




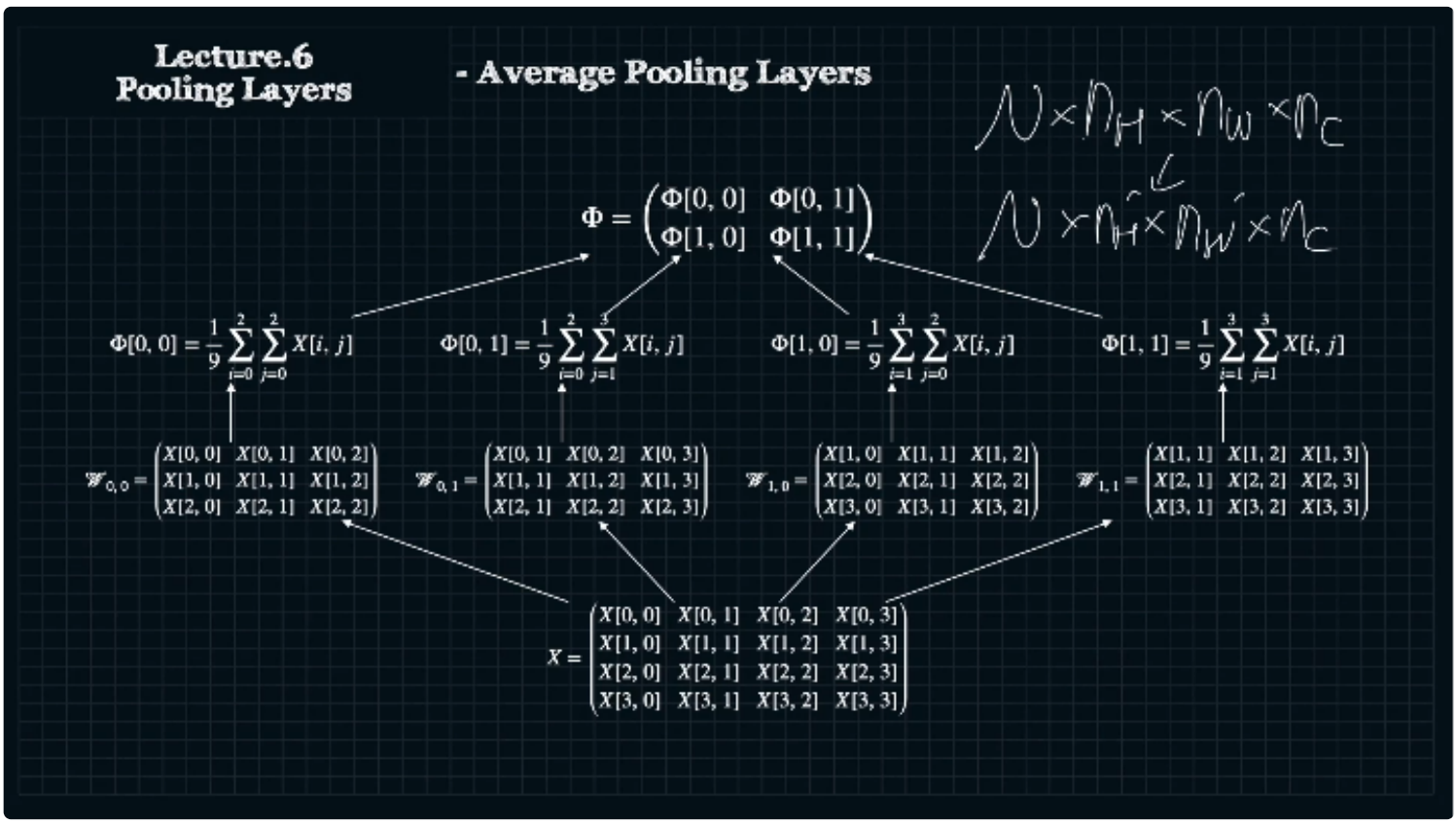
이차원에서도 같다.

Average Pooling layer



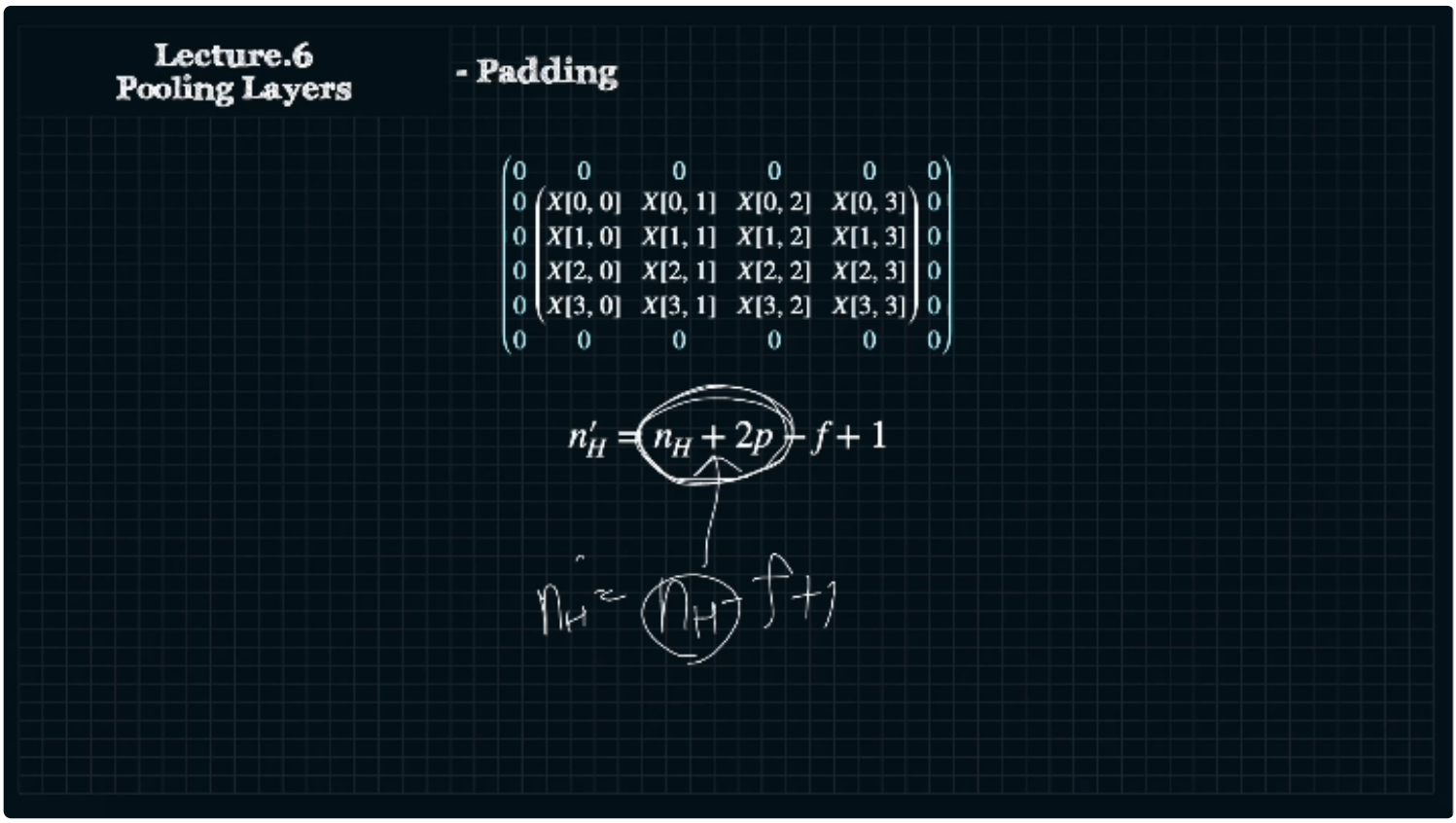


Window우 하나 뽑고 R 채널 최대값 G채널 최대값 B채널 최대값이런식으로 뽑는다.



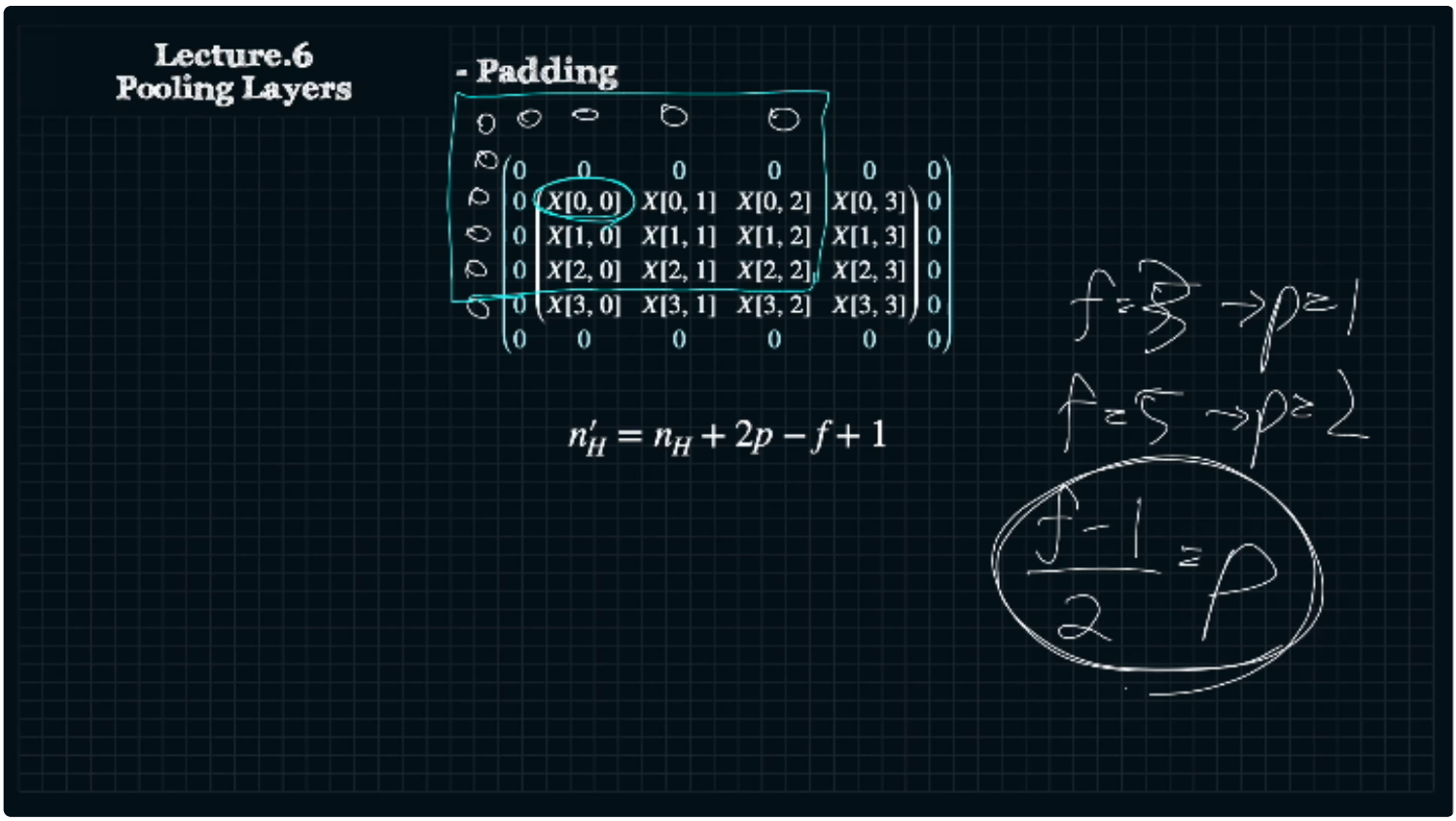
Pooling은 이미지별로 Channel wise가 아니라 이미지 별로 pooling을 해준다는 것이다.

Padding



Padding을 해주게되면 이미지 사이즈가 커지는 효과가 있다.

padding의 제일큰 장점은 input size와 output size가 달라지지않는다. height width가 달라지지 않는다.

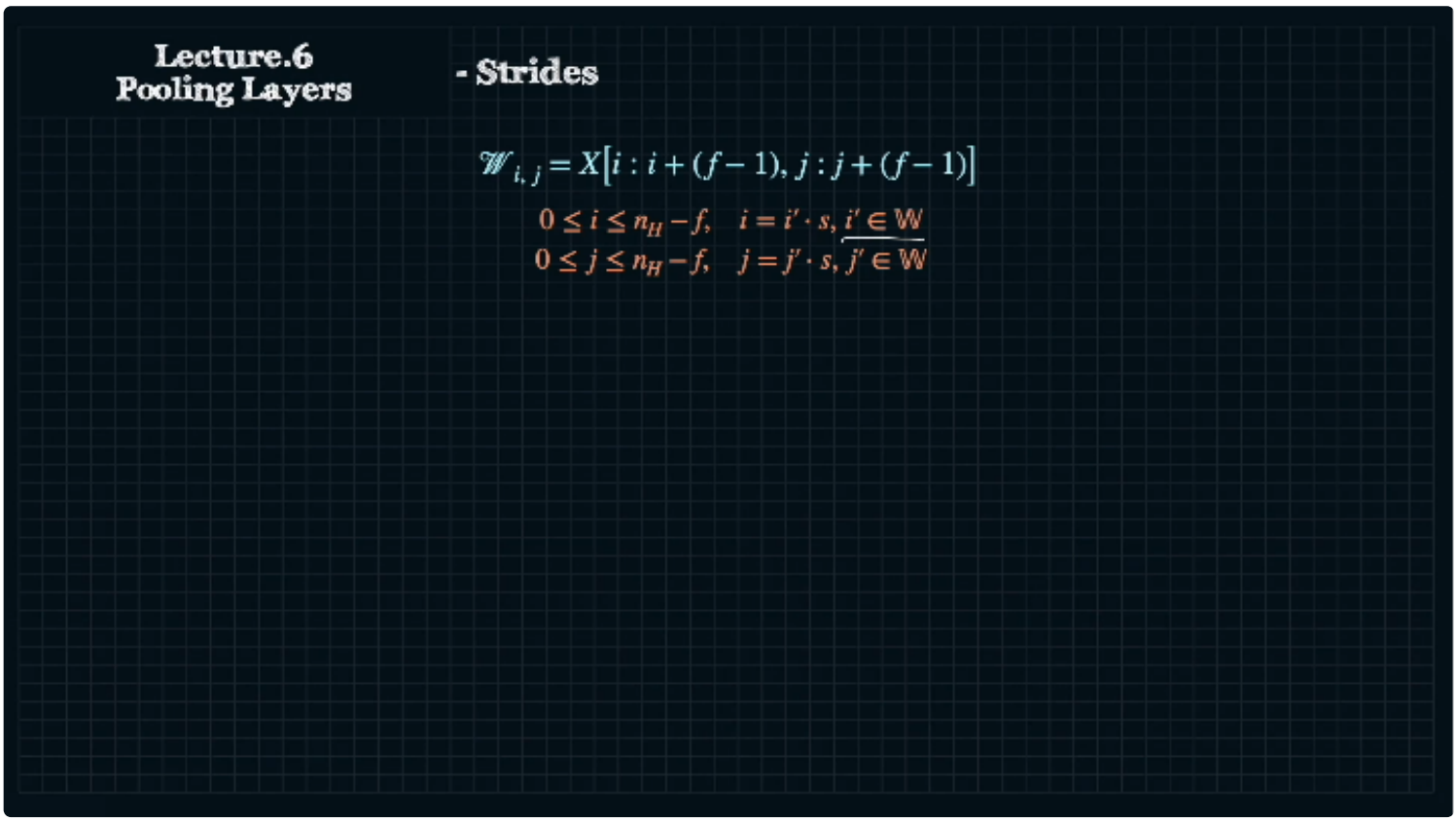


filter size = 3 padding = 1 output shape 안바뀐다.

filter size = 5 padding = 2 output shape는 안바뀐다.

Padding size 결정방법 사진 오른쪽 아래 P 공식을 사용하면 output shape는 바뀌지 않는다.

Strides



어떤 이미지가 있을 때 stride가 2면 깡충깡충 뛰는 것이다. 두칸 씩 뛰는 것이다. window를 이동할 때 깡충깡충 뛰는 것 stride가 3이면 3칸씩 이동하는 것이다.

자연수에다가 0을 포함하는걸 whole number이라고한다.

Lecture.6 Pooling Layers

- Strides

$$\mathcal{W}_{i,j} = X[i:i+(f-1), j:j+(f-1)]$$

$$0 \le i \le n_H - f, \quad i = i' \cdot s, i' \in \mathbb{W}$$

$$0 \le j \le n_H - f, \quad j = j' \cdot s, j' \in \mathbb{W}$$

$s=2$

$i = 0, 1, 2, \cdots$

$i = 0, 2, 4, 6,$

Lecture.6 Pooling Layers

- Strides

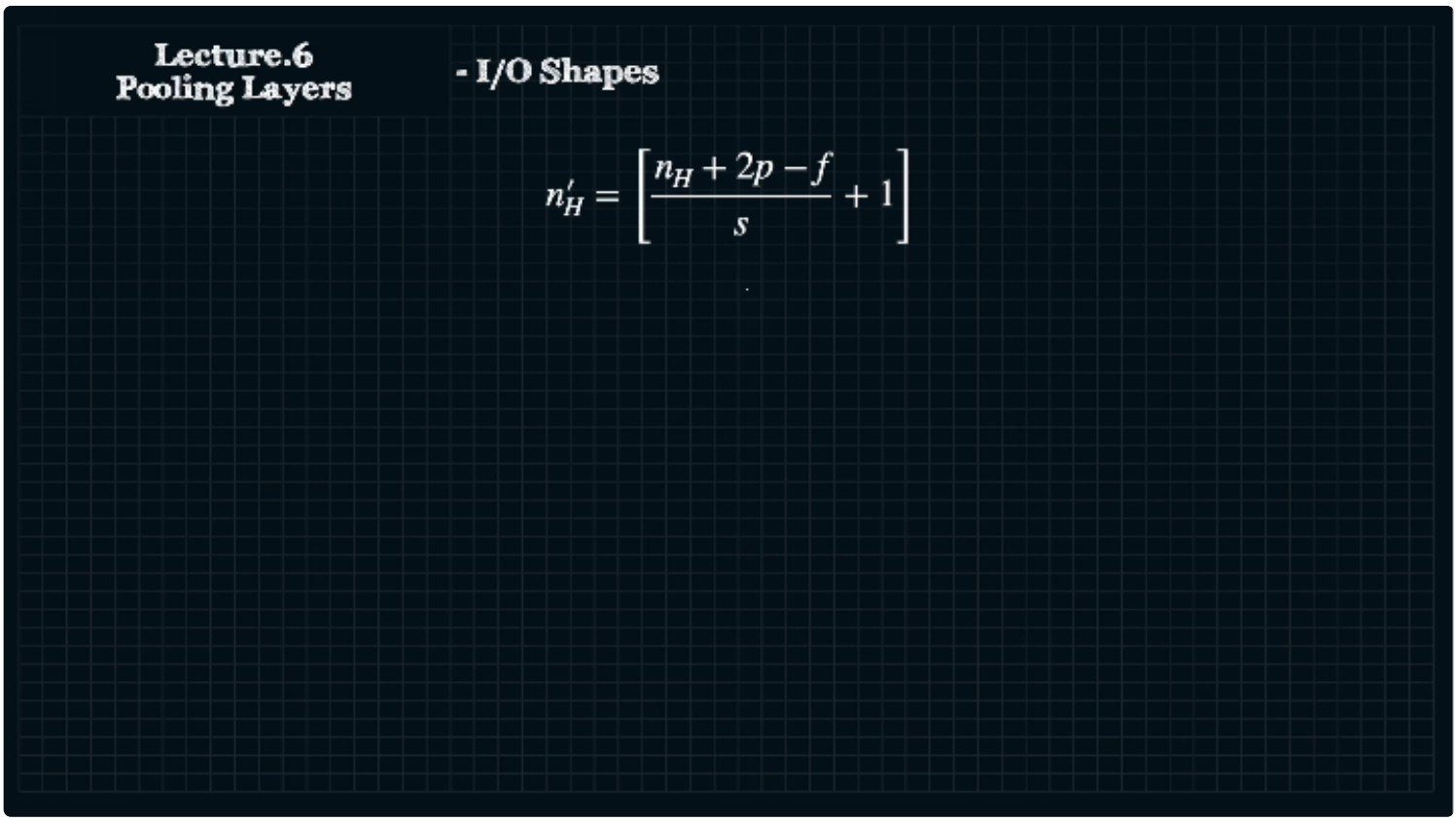
$$\mathcal{W}_{i,j} = X[i:i+(f-1), j:j+(f-1)]$$

$$0 \le i \le n_H - f, \quad i = i' \cdot s, i' \in \mathbb{W}$$

$$0 \le j \le n_H - f, \quad j = j' \cdot s, j' \in \mathbb{W}$$

$$n'_H = \left[\frac{n_H - f}{s} + 1\right]$$

가우스함수가 들어간다. 예를들어 2.5가 나오면 2로 바꿈



Convolution layer와 Max pooling layer의 차이점은 서로의 연산의 과정만 다를 뿐이지 input과 output을 만들어 내는 과정은 같다.