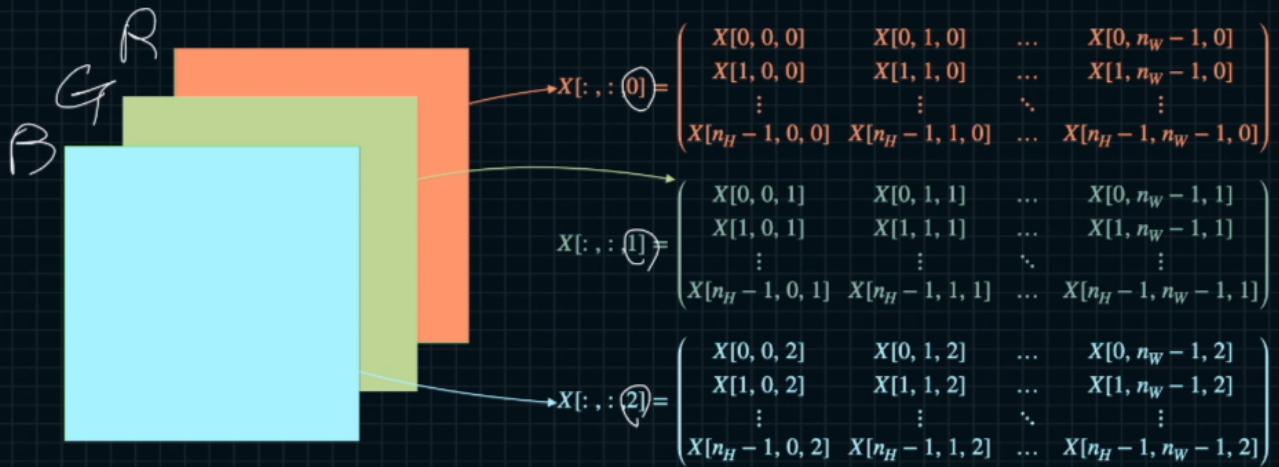


## Lecture.5 Conv Layers

### - n-Channel Input

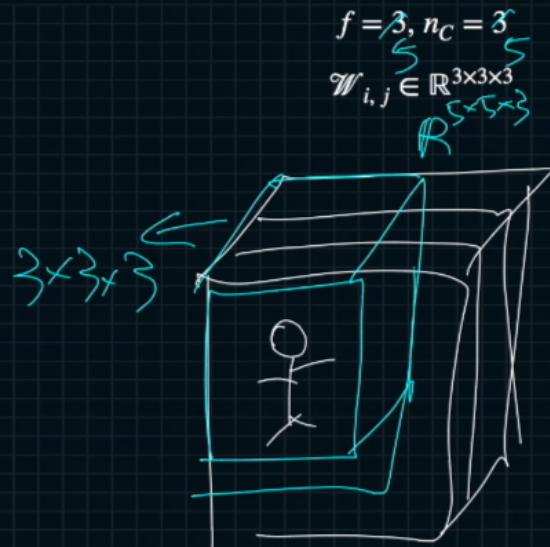
$$X \in \mathbb{R}^{n_H \times n_W \times n_C}$$



- 이차원 데이터에서의 차이점은 마지막 채널만 늘어났다고 생각하면된다.

## Lecture.5 Conv Layers

### - n-Channel Input



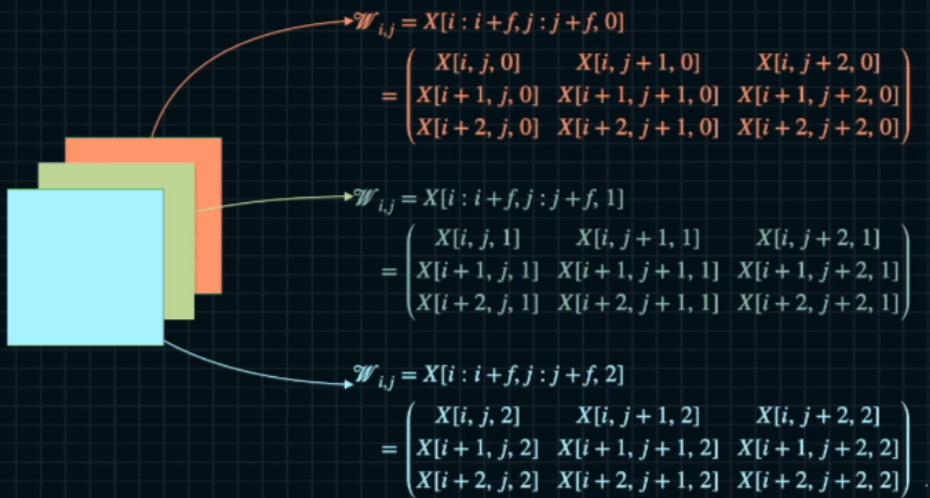
- 이전에 Conv Layer에서 filter 개념도 여기에도 적용이 가능하다. 이전에 2차원일 때는 윈도우가 움직이면서 filter 연산을 했다. 이미지가 3차원일 때도 달라지는 건 filter가 RGB값도 본다는 것 그래서 filter 차원을 표시하자면 3 x 3 x 3이다.

## Lecture.5 Conv Layers

### - n-Channel Input

$$f = 3, n_C = 3$$

$$\mathcal{W}_{i,j} \in \mathbb{R}^{3 \times 3 \times 3}$$



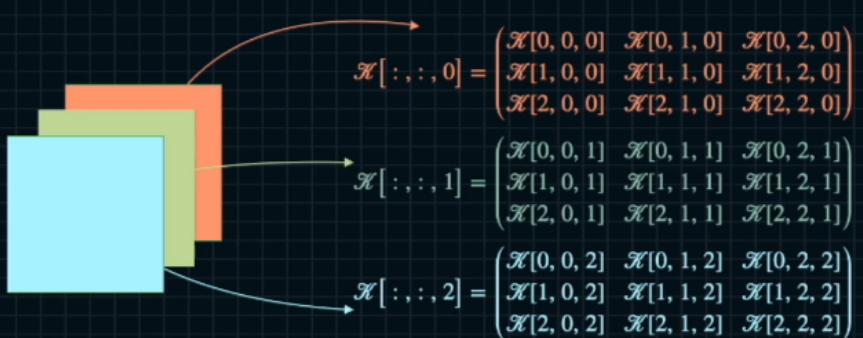
- 자세하게 보면 이렇게 RGB값들이 window로 같이 뽑히게 되는것이다.

## Lecture.5 Conv Layers

### - n-Channel Input

$$f = 3, n_C = 3$$

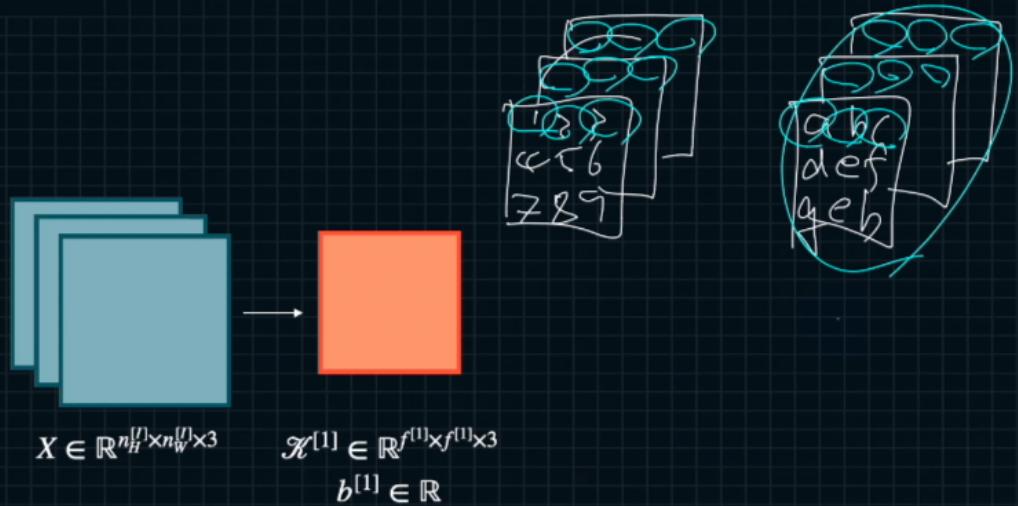
$$\mathcal{K}_{i,j} \in \mathbb{R}^{3 \times 3 \times 3}$$



- window가 뽑히면 filter도 같이 나온다.

## Lecture.5 Conv Layers

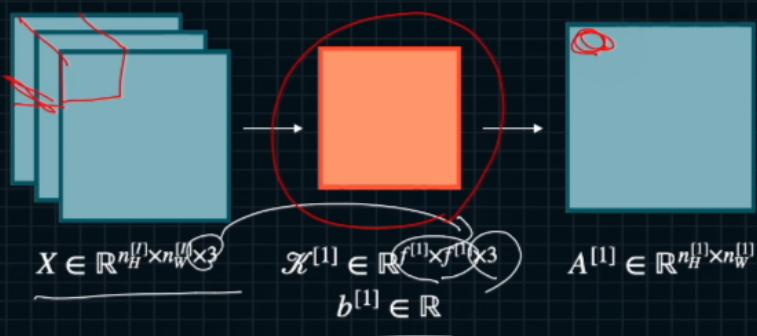
### - Conv Layers



- window에서 뽑힌 값들이랑 filter랑 연산하는건 2차원일 때랑 같다 차이점이 있다면, 차원이 늘어난것이다.

## Lecture.5 Conv Layers

### - Conv Layers

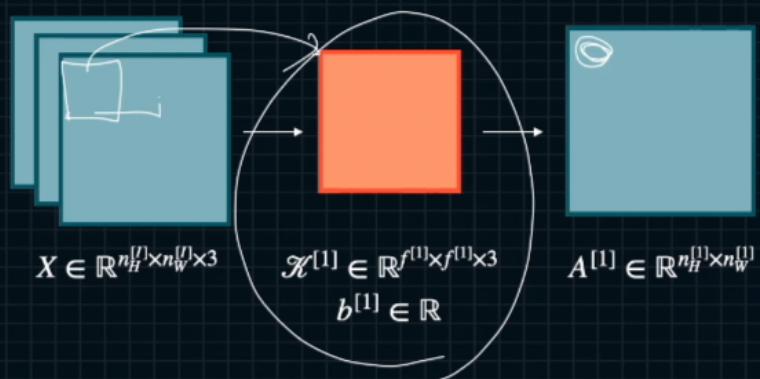


- 3차원 window 와 filter가 연산이 끝나면 오른쪽 마지막 그림처럼 하나가 뽑히는것이다. 하나의 스칼라 값이 뽑을 수 있음
- 채널이라는 depth가 생긴다.

## Filter bank

### Lecture.5 Conv Layers

#### - Conv Layers

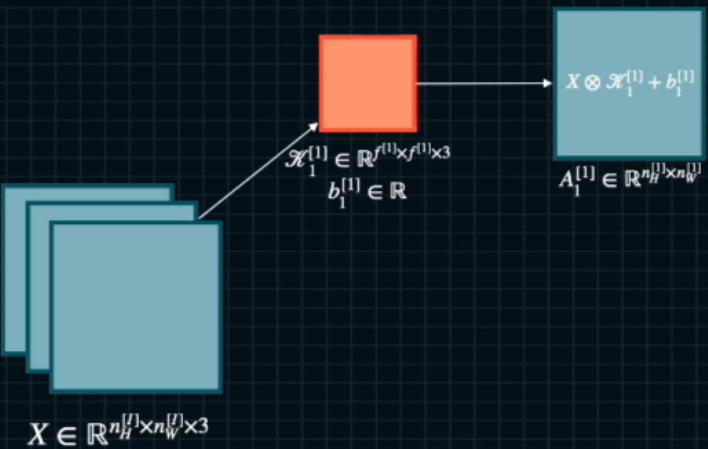


Filter가 왼쪽부터 오른쪽 밑에까지 Filter와 window와 연산을 하게된다.

딥러닝에서는 Filter를 여러개 모아둔 Filter bank를 사용하게 된다.

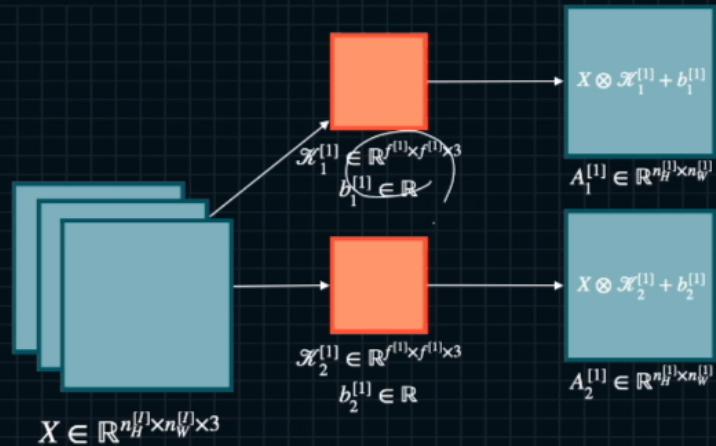
### Lecture.5 Conv Layers

#### - Conv Layers



## Lecture.5 Conv Layers

### - Conv Layers

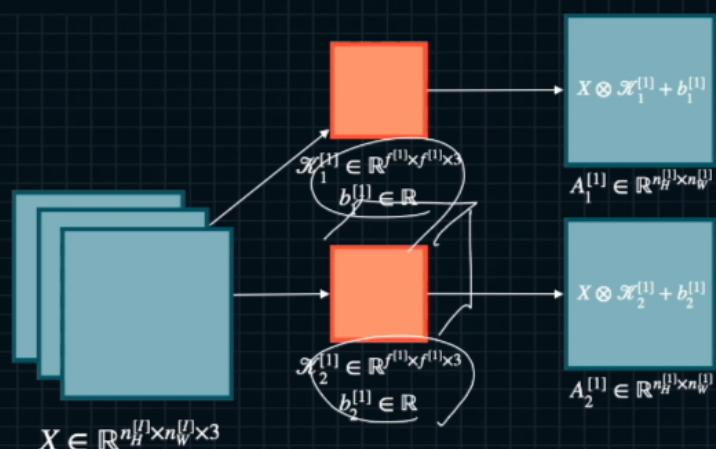


첫 번째 필터 연산과 두 번째 필터연산에서는 Weight와 bias는 똑같다.

예를들어서 Dense layer에서 뉴런들이 존재하는데, 뉴런의 개수 표현은  $1 \times L_i$ 개로 표현이된다. 뉴런안에 들어있는 weight도  $L_i$  만큼 존재한다. 이거랑 동일하다. 여기서도 Filter가 늘어난거 뿐이지 Weight와 bias의 모양은 동일하다. 하지만 달라지는 점은 Weight와 bias의 모양이 같더라도 안에 들어있는 값들이 다르다.

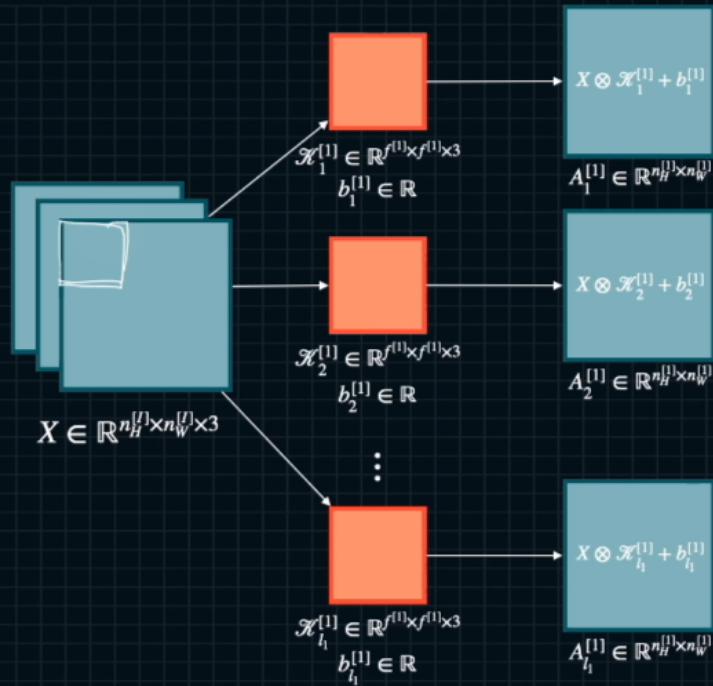
## Lecture.5 Conv Layers

### - Conv Layers



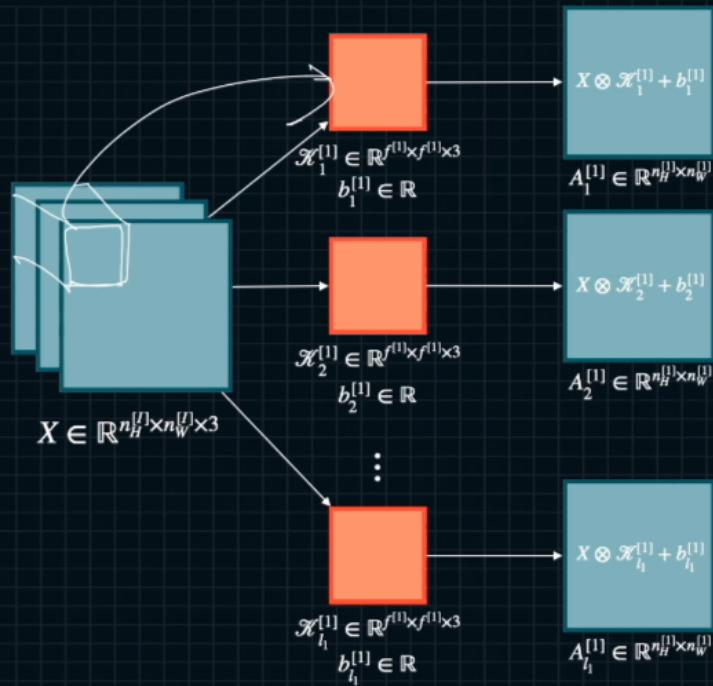
## Lecture.5 Conv Layers

### - Conv Layers



## Lecture.5 Conv Layers

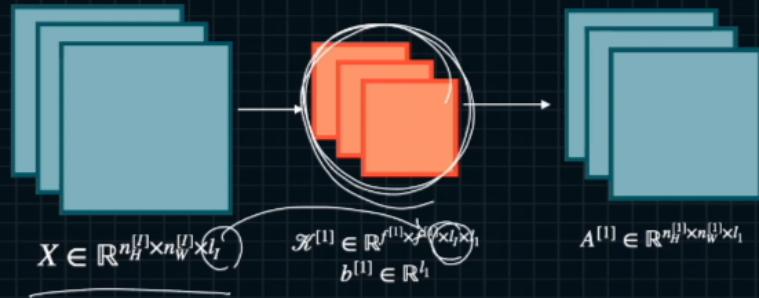
### - Conv Layers



3차원 이미지에서 첫 번째 filter와 연산을 하게되면 하나의 이미지가 만들어진다. 그래서 애네는 다른 correlation 값을 가지고 있기 때문에 각각의 다른 이미지들을 얻을 수 있다. matrix를 얻을 수 있다.

## Lecture.5 Conv Layers

### - Conv Layers



input channel이  $l_i$ 개 들어왔으면, Filter도  $l_i$ 만큼 가지게 된다.

bias는 kernel당(Filter) 하나씩 가지게 된다.

## Lecture.5 Conv Layers

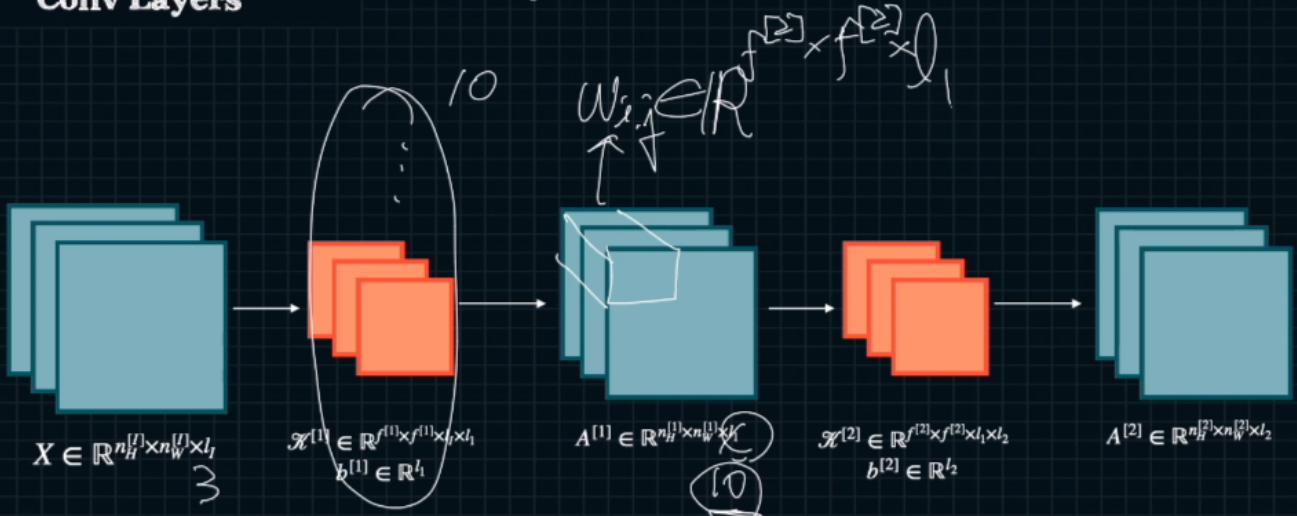
### - Conv Layers



$l_i$ 차원을 가진 이미지가 인풋으로 들어오고  $l_i$  만큼 filter가 생긴 후 이 filter와 이미지에서 같은 window값들과 correlation 연산을 하게 된 후 activation을 통과하게 되면 하나의 이미지를 만들어 낼 수 있다. 그리고 또 다시 이 이미지가 input이 되는 것이다.

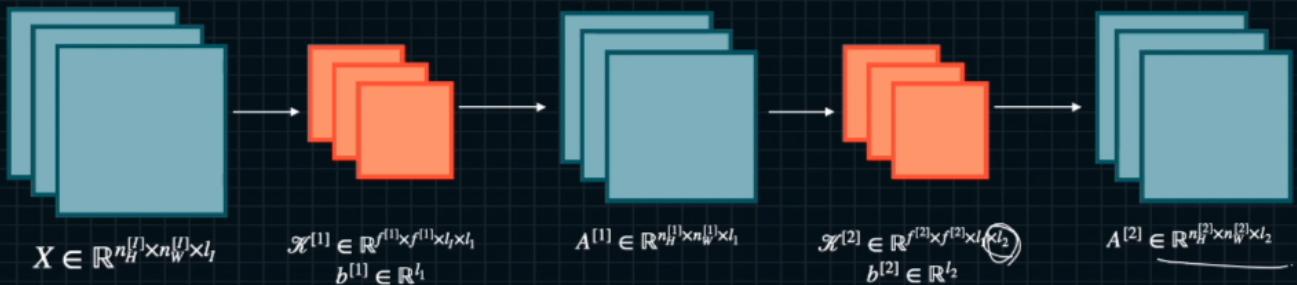
## Lecture.5 Conv Layers

### - Conv Layers



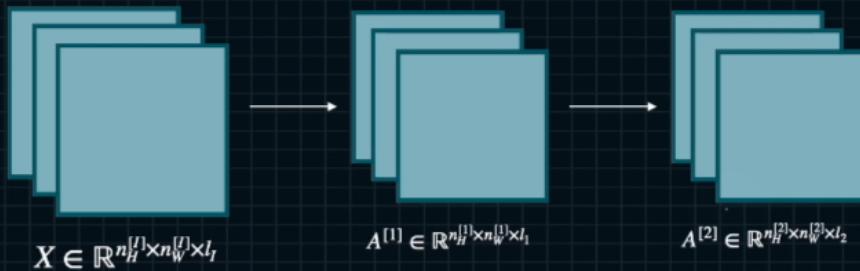
## Lecture.5 Conv Layers

### - Conv Layers



## Lecture.5 Conv Layers

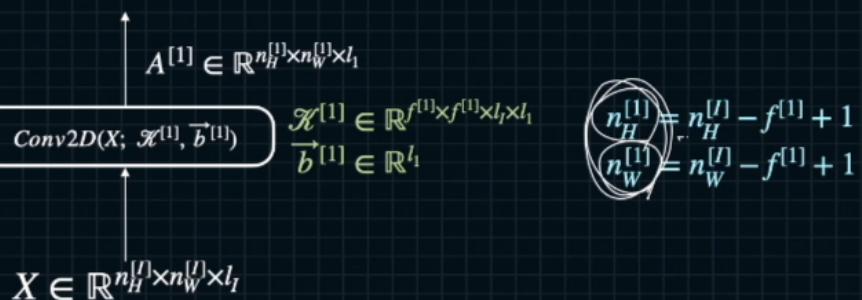
### - Conv Layers



## Cascaded Conv Layers

### Lecture.5 Conv Layers

### - Cascaded Conv Layers

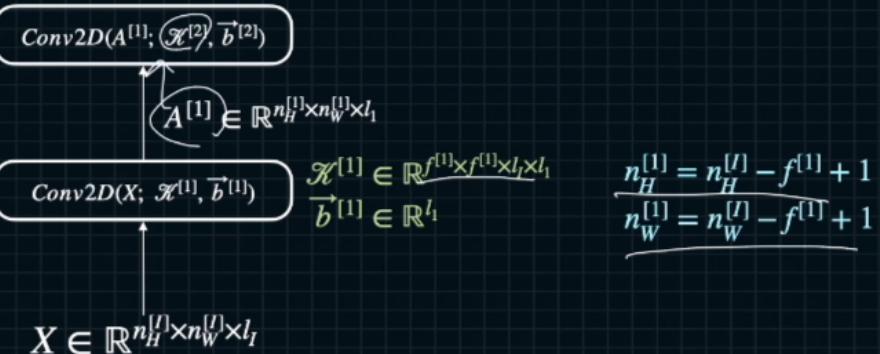


이제 그림 설명이 아니라 수식으로 짚여하자면  $X \rightarrow$  이미지 이미지가 Height x Width x channel 만큼 input이 들어가면 Conv에서는  $F \times F \times \text{channel} \times \text{Filter count}$  즉, input image 때문에 channel만큼 가지게 되는것이고 그리고 filter가 몇개 쌓일 것인지가 Filter

count다 필터 count만큼 bias를 가지게 된다. 그리고 이 결과가 activation function을 통과하고 image size는 초록색 부분을 보면된다.

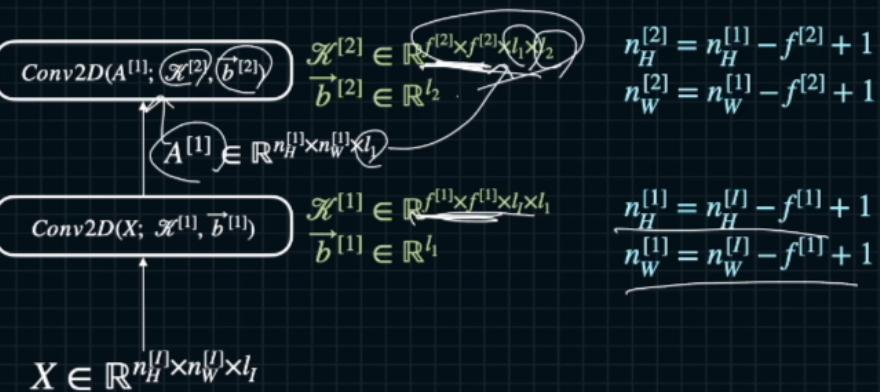
## Lecture.5 Conv Layers

### - Cascaded Conv Layers



## Lecture.5 Conv Layers

### - Cascaded Conv Layers



여기가 일반적인 conv layer의 연산이다. 여러개의 이미지가 들어온다고 해서 weight 와 bias는 변하지 않는다. minibatch 개념

## Lecture.5 Conv Layers

### - Minibatch in Conv Layers

