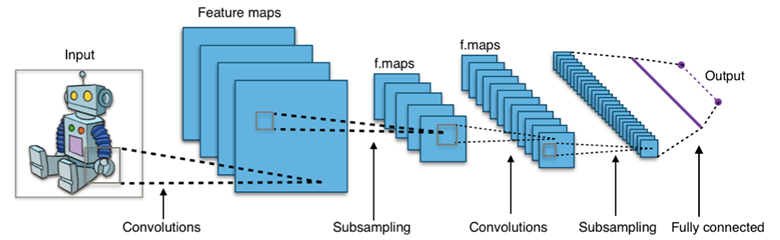
About CapsNet

# Typical CNN



Why subsampling 🡪 one by one mapping out is inefficiency

## Concept of pooling

* Creates “summaries” of each sub-region
* Give **positional and translational invariance**
* Classification
* 각각의 sub-region 을 요약하고, 위치 정보에 invariance 를 부여하고 분류한다.

# Problem of CNN

## Limit of Convolution filter

Convolution filter detect what’s important in an image pixel.  
Low level (simple feature - edge, color), High level (complex feature), Top level layer(Classification)  
But the biggest problem with **higher layers is simply the weighted sum of the lower layers**.  
This weighted sum does not consider the positional relationship between simple and complex feature at all.

컨볼루션 필터는 이미지 픽셀에서 중요한 부분을 디텍팅한다. 낮은 단계에서는 단순한 특징(모서리, 색깔), 높은 단계에서는 복잡한 특징, 가장 상위 레이어에서는 분류를 합니다. 문제는 최상의 레이어는 단순히 낮은 레이어의 가중합으로 구성된다. 이런 가중합은 낮은 단계의 특징과 상위 단계의 복잡한 특징간의 위치 관계를 전혀 고려하지 않는다는 것이다.

## Max-pooling

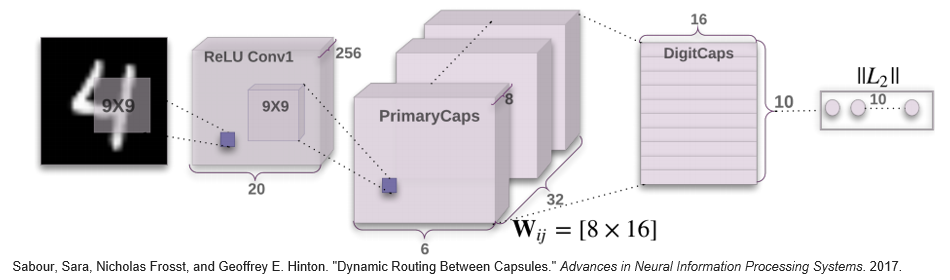


CNN은 위에 문제의 해결을 위해 Max pooling 이라는 방법을 사용합니다.

각 feature map 은 spatial size 를 줄여줘 higher layer의 시야각(field of view)를 넓혀주는 역할을 한다. 위 그림에서 FOV값이 작으면 노란색 직사각형이 사용자의 시야가 되는 것이고, FOV값이 크다면 파란색 직사각형 이상의 시야가 되는 것 입니다.

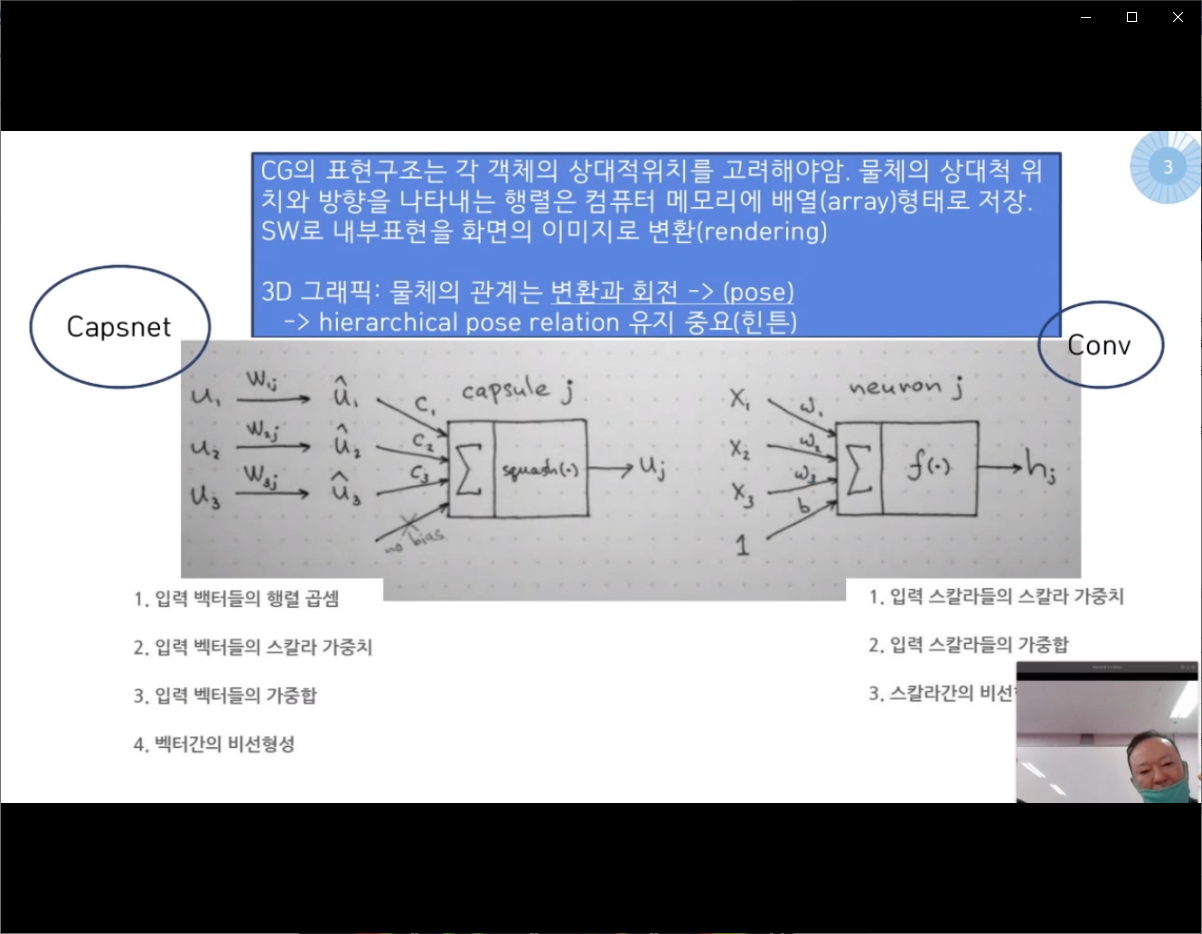
이로 인해서, higher level 의 시야가 넓어지고 spatial 한 정보를 일부 가져가지만 근본적인 해결책이 되지는 못했고, feature detector의 위치 정보를 간과하고 있습니다.

# Talk about CapsNet architecture



익숙한 예제인 MNIST를 가지고 구조를 관찰하면 다음과 같습니다.

* Input : 28 x 28 x 1
* Conv1 kernel : 9 x 9 x 256, stride 1  
  Output : 20 x 20 x 256
* Primary Caps : 9 x 9 x (32 x 8), stride 2, **no activation use squash function**
* Digit Caps : dynamic routing



이 화면에서 살짝 헷갈린 부분이 Caps net의 내용 이였는데,

논문에 따라 정리해보면 다음과 같습니다.

은 Conv1 Layer의 Output (20 x 20 x 256) 을 의 Conv 8D(9 x 9 x 32 stride of 2) 한 결과입니다.

결과적으로 Primary Capsules 은 [32 x 6 x 6 ] 의 output 이며 각 output 은 8D vector입니다.

이것이 의 정체였습니다.

즉, 코와 입 눈 등등 지엽적인 부분은 Convolution을 통해서 확인하고, Caps Net으로 넘어가게 됩니다.

다음화에 계속