

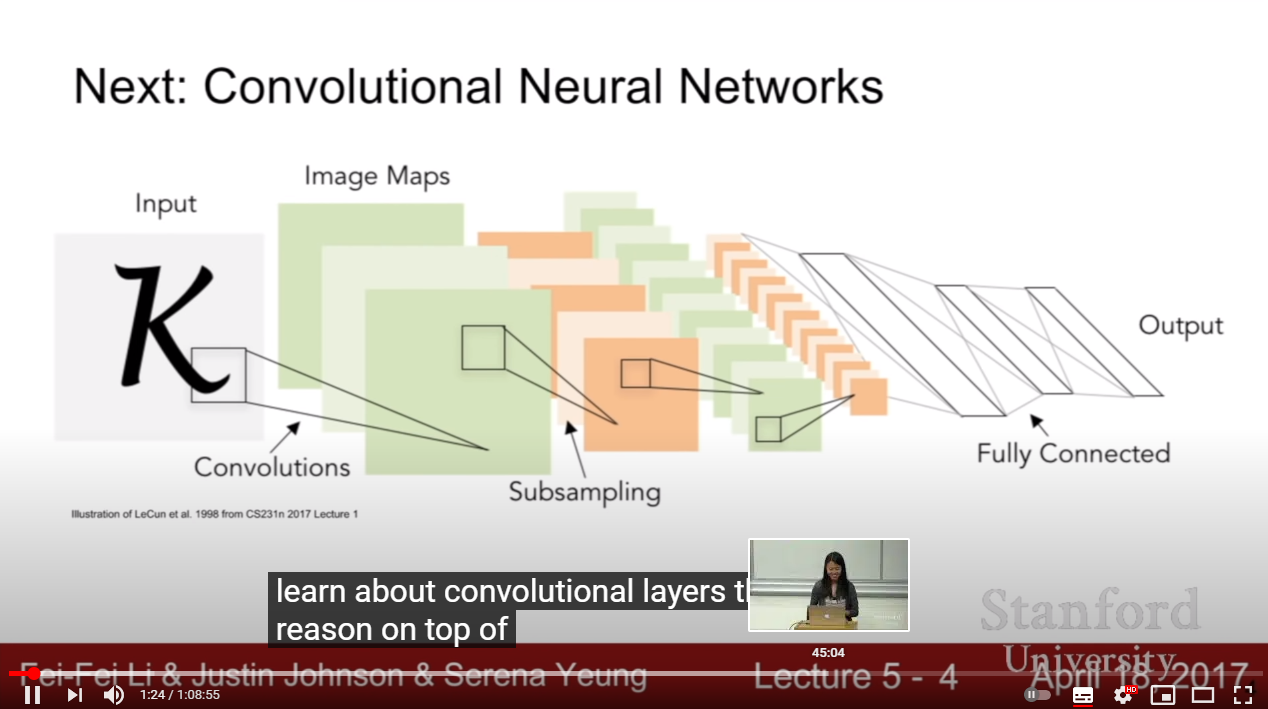
우리는 지난시간에 f=Wx라는 linear형태의 score function(x는 input, W는 가중치)을 보았으나

이제는 f=W2 max(0,W1x)라는식의 2-layer NN을 볼 것이다.

예를들어 처음에 3072개의 x가 들어온다면, x에 대한 첫번째 가중치 w1과 곱해져서 h라는 숨겨진 레이어가 추가된다. 이 때 h는 x의 w1에 대한 score이며, w2는 이 h에 대한 가중치이다.

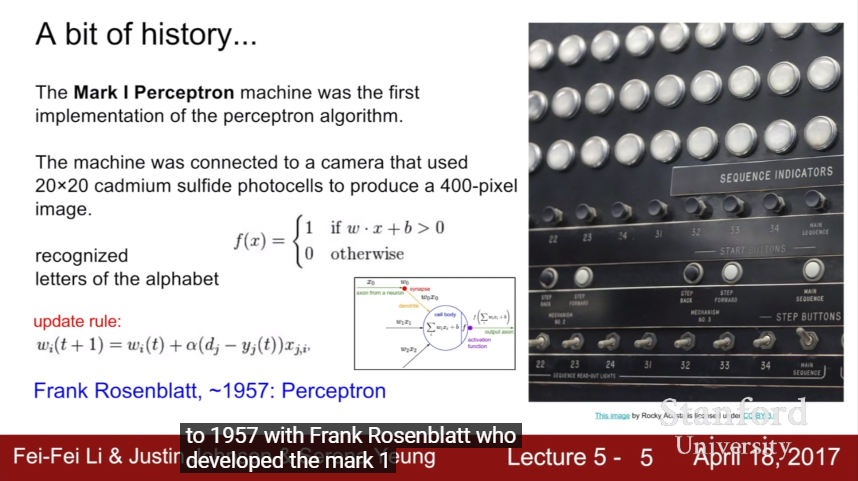
이 히든 레이어는 일종의 분류기 역할을 하게되는데, 히든 레이어가 많다면 분류기가 많다는 뜻이 된다.

이번에 배우고자 하는건 CNN이다.

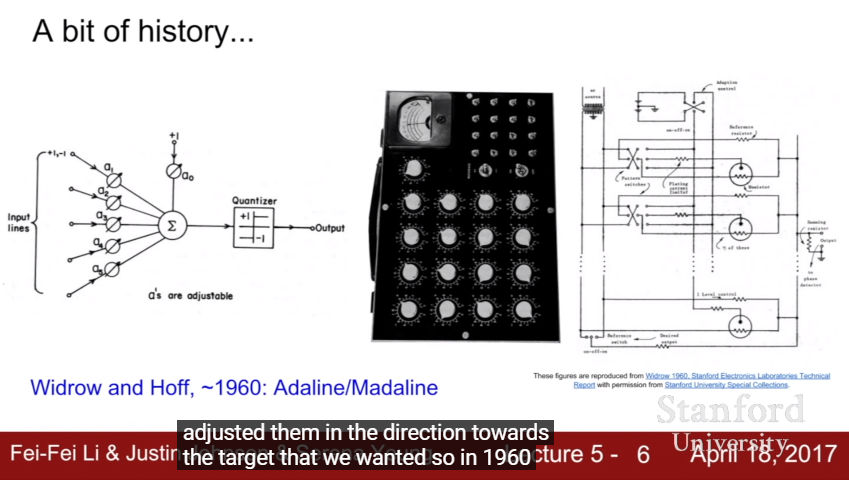


이 CNN을 배우기 앞서서 딥러닝의 역사를 한번 살펴볼 예정이다.

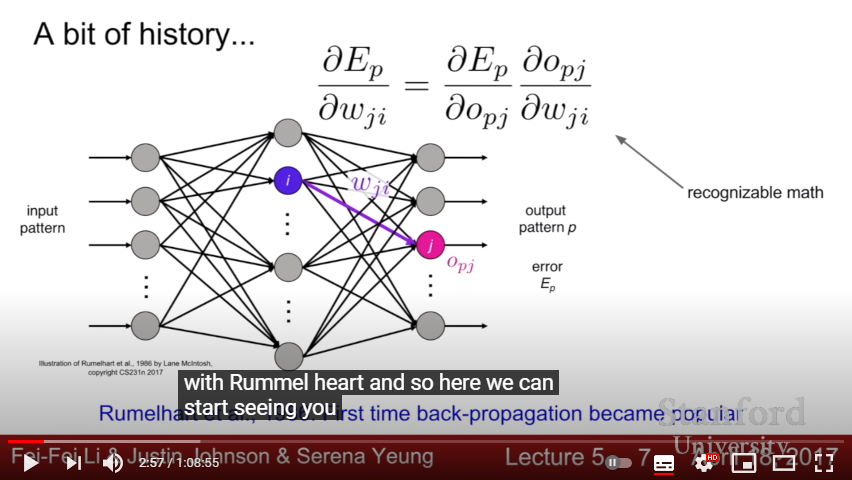
수학이 안나와서 어렵진 않은데 뭔 말인지 알아먹는게 어려웠다



이 기계는 mark 1 Perceptron이란건데, perceptron은 두뇌의 방식을 모방한 인위적 네트워크를 말한다. 즉,,,~~최초의,,인공,,신경망,,기계,,인,,것이다,,,~~!!! 57년생이면 우리 아빠보다 형이다

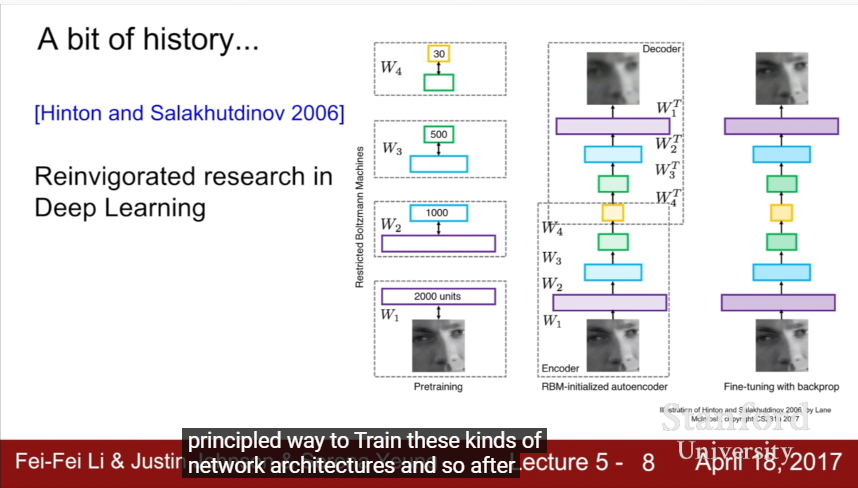


요것은 2세대 기계이다.

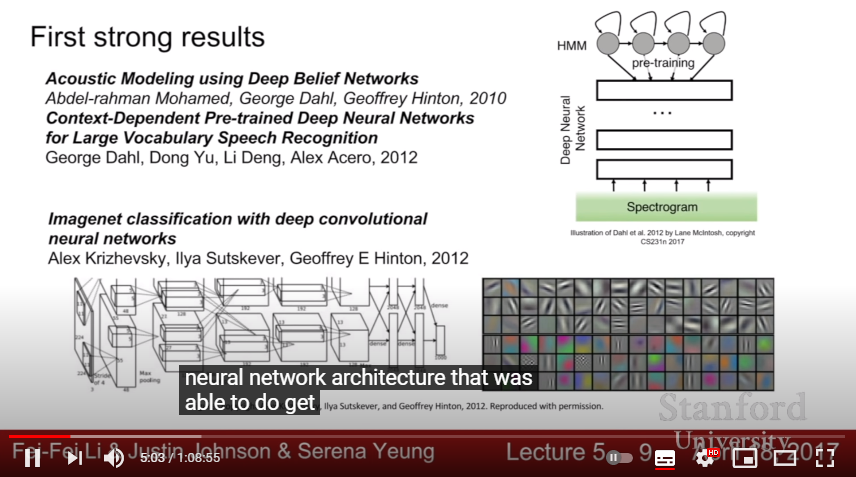


4강에서 나를 괴롭혔던 back propagation은 놀랍게도 내가 태어나기 전엔 1986년에 등장한다.

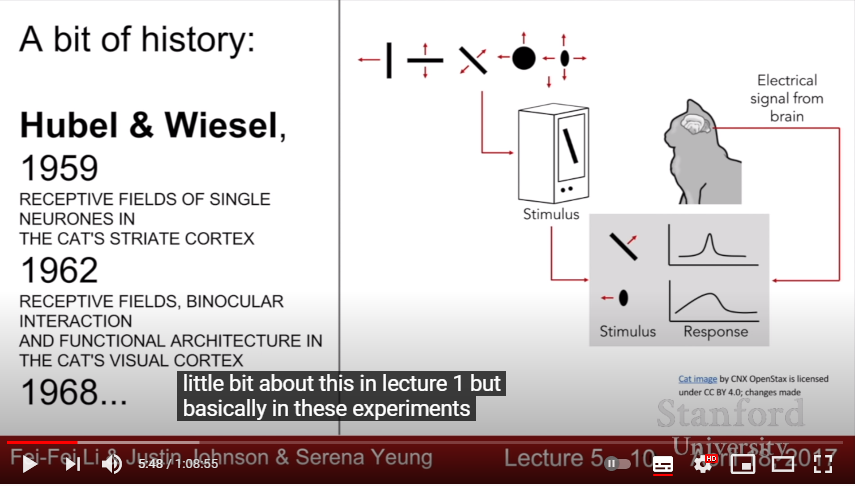
참고로 이 때도 인기는 없었는지 한동안 언급이 안되었으나…



잔짜잔 Hinton & Salakhutdinov(살라쿳디노프?) 센세가 재조명을 하게되고

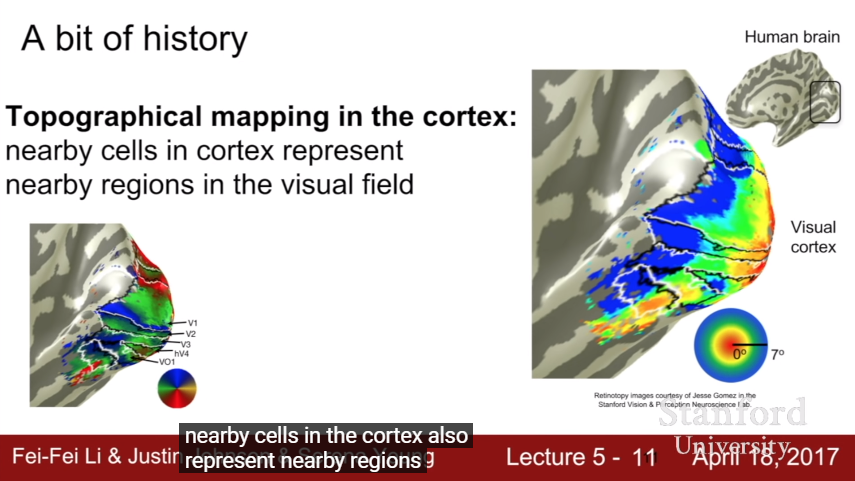


드디어 2012년에 CNN을 활용한 이미지 인식률이 떡상하여 빛을 보게 된다

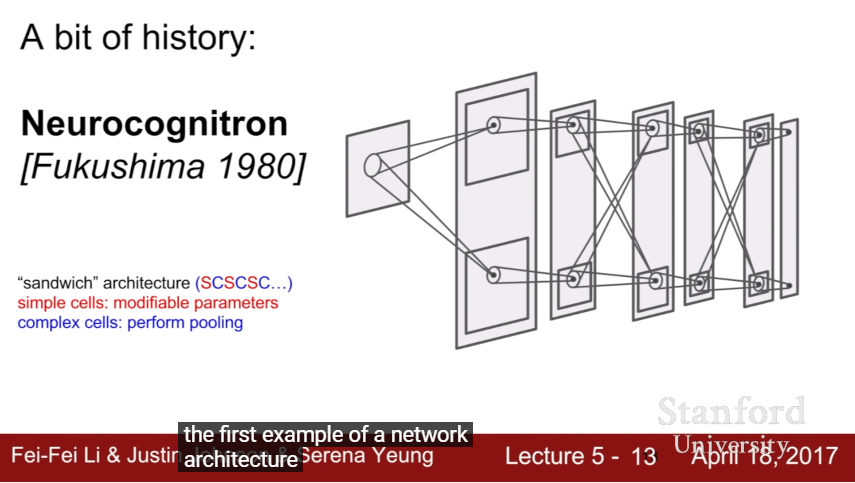


인공신경망과 별개로 CNN의 역사를 다시 보자면,

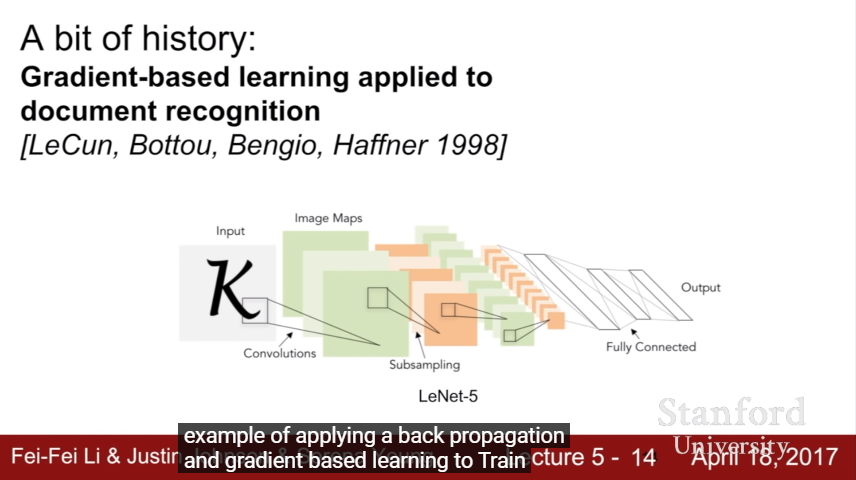
애옹이 두뇌를 활용해서 이미지 인식의 방식을 연구한데서 시작하였다



이는 사람도 마찬가지로, 뇌에서 시각을 담당하는 두뇌에서 영감을 받았는데, 시각에 의해 두뇌가 활성화되는 것을 보고 이를 적용한 것이다



후쿠시마? 아무튼 우리가 이전강의에서 본것과 유사한 컨셉이 1980년에 이미 이슈가 된 것을 볼수있고



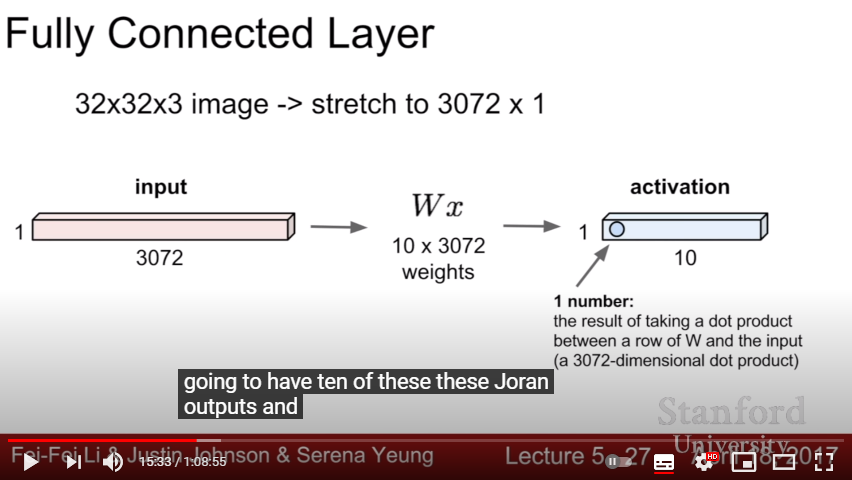
LeCun센세가 CNN이라는 개념을 정립하게된다(1998년)

이후 이것이 발전하여 AlexNet 등이 되고 이전에 보았던 이미지 인식 테스트에서 인식률 떡상이라는 결과로 나타나게 된다.



그 결과 사회의 다양한 분야에서 이미지 학습을 통한 Visual Recognition이 이루어지고 있다

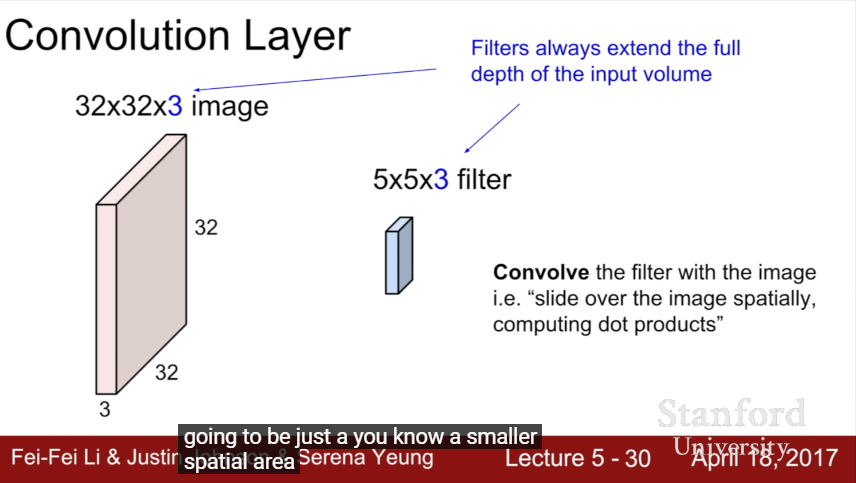
그래서 우리는 드디어 이 강의의 핵심인 CNN에 대해 배워볼 예정이다.



초기엔 32\*32\*3 사이즈의 이미지(3은 RGB)가 있다면, 이를 1\*3072로 Stretch하여 f=Wx에 넣고 결과를 확인했다.

그러나 이는 그렇게 좋은 방법이 아니다.(왜? 이해를 못한건가)

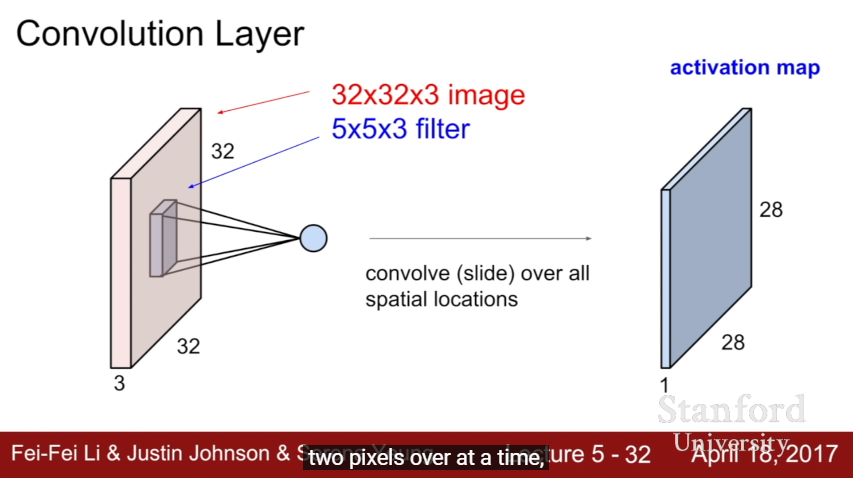
=> Parameter 수 // **spatial invariance // 3차원 -> 1차원 공간정보 손실이 발생**



그래서 이젠 방법을 바꿔서, 원본의 32\*32\*3 이미지를 그대로 두고,

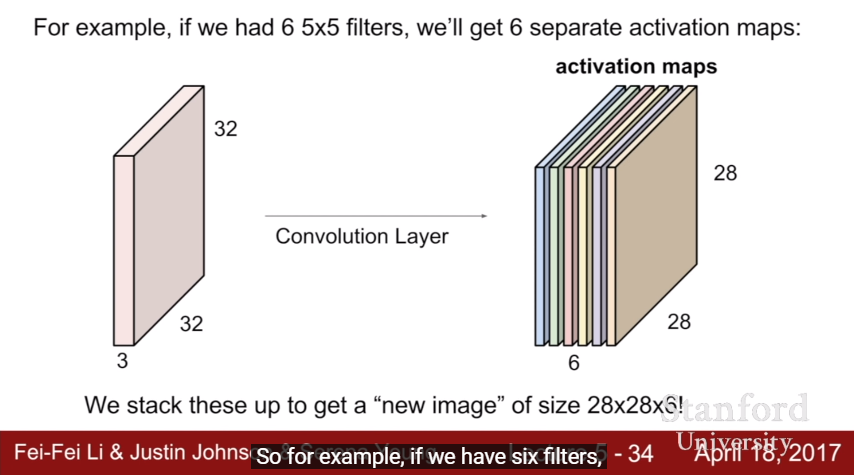
5\*5\*3 사이즈의 필터를 준비한다.(필터와 가중치 w의 상관 관계?)

이 필터는 dot product를 계산하면서 이미지 위를 [공간적으로] 움직이게 된다.



그 결과, 28\*28\*1의 결과가 나온다 이를 Activation map이라고 부른다.

이 Activation map은 5\*5\*3이라는 filter가 걸러낸 특징을 가지고있는 하나의 층이다.



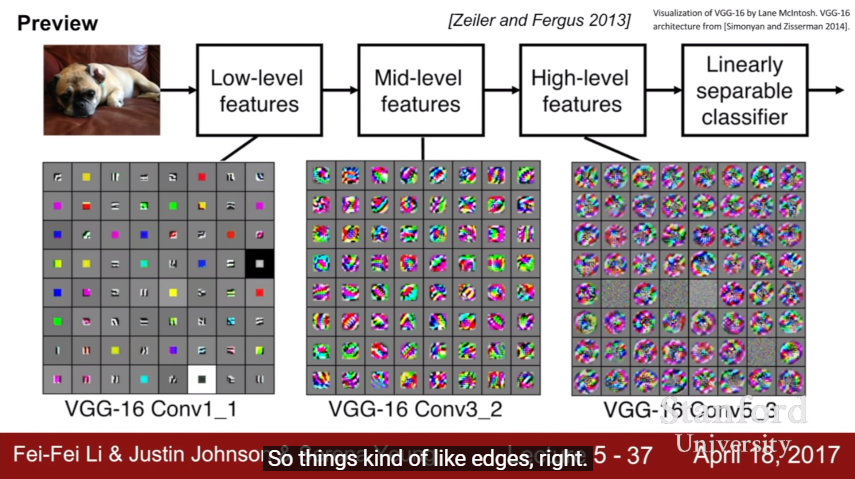
이때 필터를 여러 개 쓴다. 위는 그런식으로 총 6개의 다른 필터를 사용한 결과이다.

RGB로 인해 3이었던 depth가 6이 되었다.



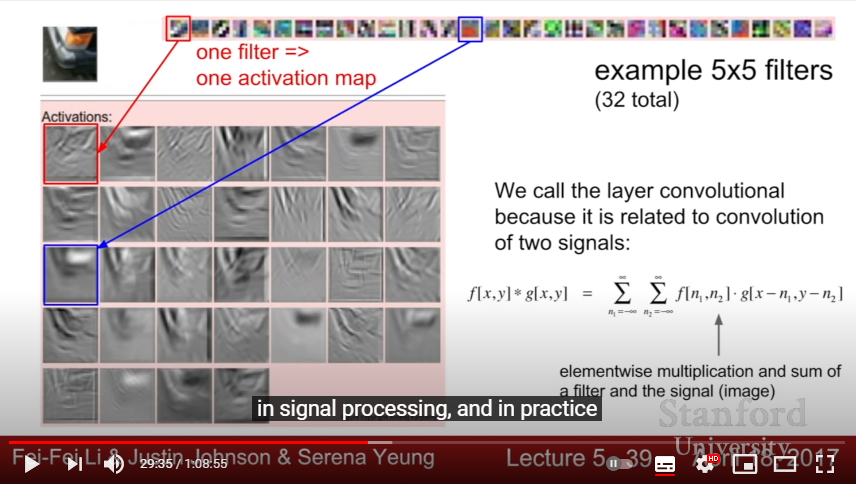
이제 depth가 6이 되었으니 필터 depth도 6이 되어야 한다.

5\*5\*6 size의 filter를 10개 쓰면 이제 24\*24\*10의 새로운 Activation map들이 생긴다. 그리고 이를 반복한다



반복하다 보면 이렇게 된다. 예전에는 말대가리가 두개거나 빨간색 자동차거나 이런식으로 뭔가기존 사진의 특징이 있었느데 이건 전혀 댕댕이같이 보이진 않고 정신나갈것같아

이는 단순히 색상뿐만이 아니라 edge라던가, 방향이라던가 하는 필터에 의해 선별된 정보가 반영되어있어서로 보인다

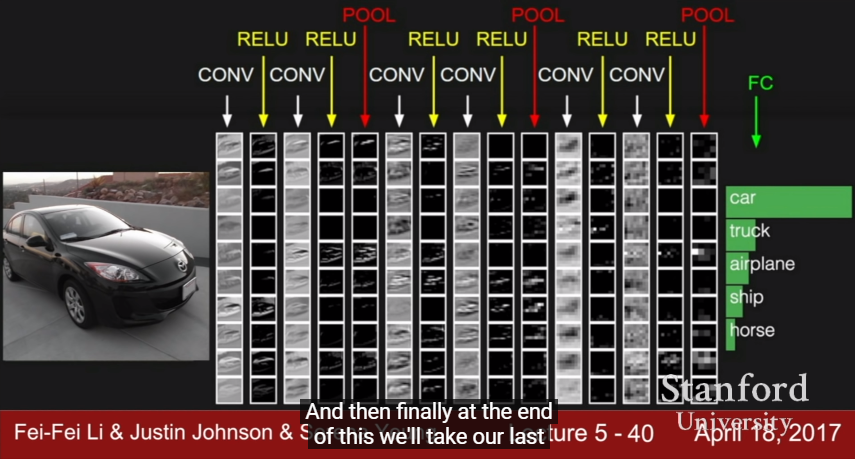


위에 보이는 이미지  이게 필터고, 이게 Activation map이다.

정확한 이해는 어렵지만, 저 파란색으로 선택된 필터의 경우 자동차의 비상등쪽 노란부분에 반응반응한 것 보아 노란색을 찾아내는 필터가 아닐까 싶다

맨 처음 빨간색으로 선택된 필터는 대략적인 모양이 위 사진의 egde와 비슷하기 떄문에 edge를 찾아내는 필터가 아닐까

아무튼 이런식으로 각 필터는 자신들이 보고싶은 이미지의 특성을 필터링하여 추출해낸다.

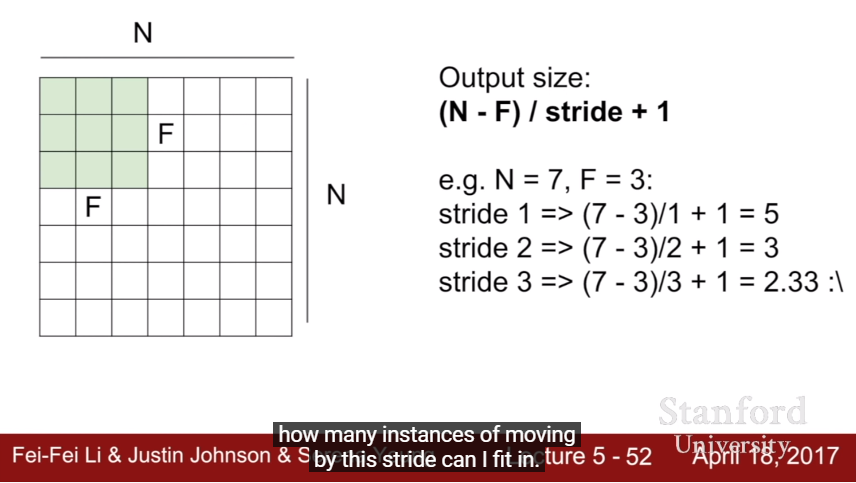


짜잔 이런식이다.

q. pool이 무엇인지

q2. 위에 댕댕이 사진은 칼라풀했는데, 필터도 칼라풀한데 왜 Activation map은 까만색인지

* 32\*32\*3이 28\*28\*1이 되면서 3이었던 RGB의 정보가 모두 사라지고 1(흑백)이 돼서??
* 맞다면 왜 위에 댕댕이의 Activation map은 칼라풀한지



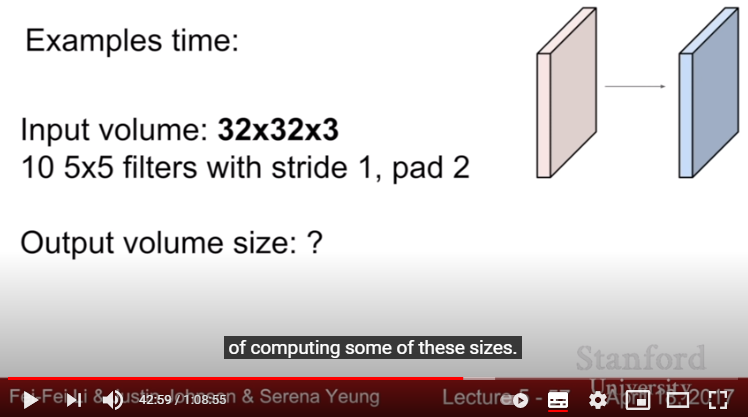
이건 필터의 사이즈에 대한 내용인데,, 앞부분은 저번에 실버 신지드 장인 Yann LeCun님이 설명해주셔서 어떤식으로 [공간적 이동]을 하는지는 패스하고, 필터의 사이즈를 정하는 방법에 대한 내용이다.



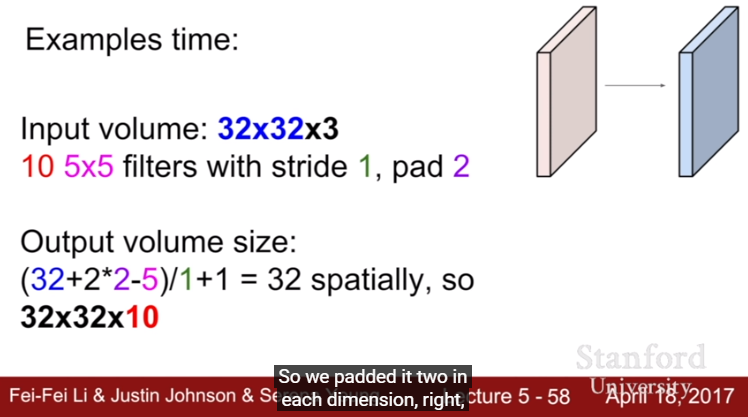
근데 저렇게 하다보면 몇층 내려가다보면 이미지 size가 점점 작아저 계산이 안되게 된다.

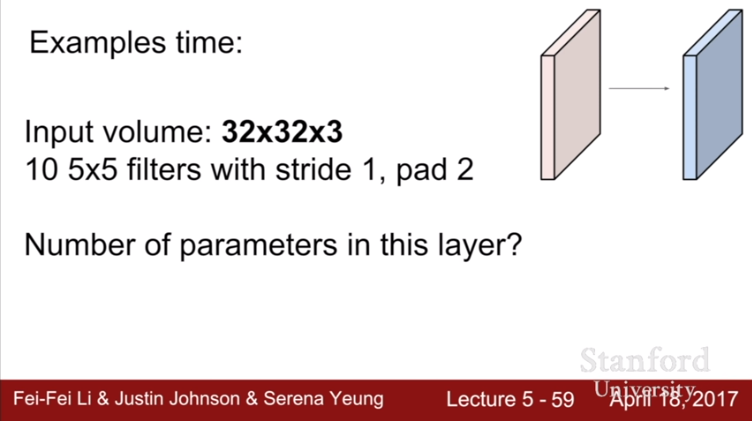
그래서 이를 극복하기 위해 겉(border)에 0이라는 이미지를 덧대어 이미지 사이즈를 맞추게 된다(zero pad)

즉, 사이즈를 유지하기 위해 사용하는것이 zero padding이다.

문제 - 32\*32\*3 input을 10개의 5\*5 필터를 (5\*5\*3?) 쓰고 여기에 stride 1, pad를 2개 붙인다면?

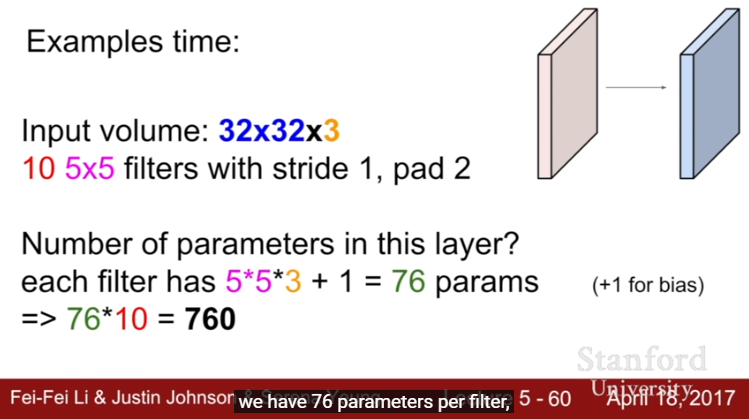
-> 32\*32\*10 size가 된다.

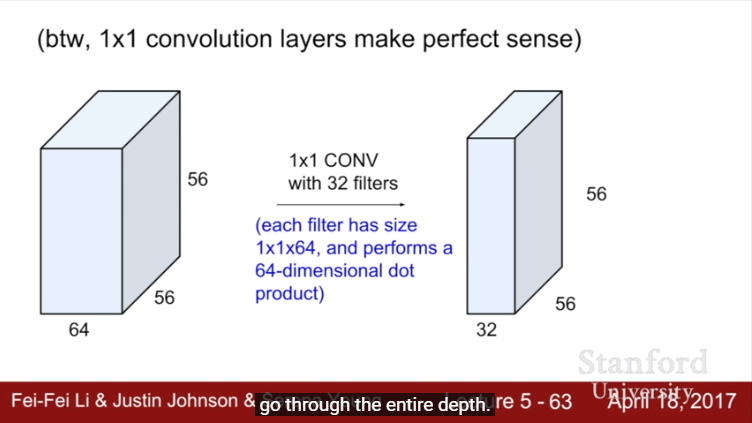




두번째 문제 - 이 때 파라미터 개수는?

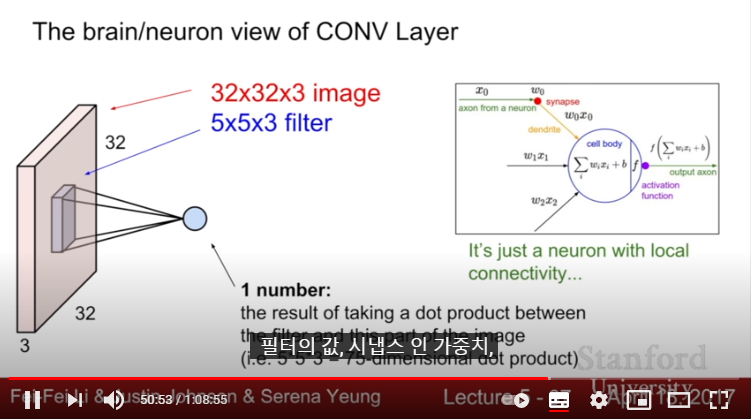
-> 이거 이해안됨





이것은 1\*1 convolution인데, filter가 1\*1이면 의미가 없는것 아닌가?? 이건 잘 이해가 안됨

-> 다시 읽어보니 의미가 없다기 보다는.. 사이즈 변화가 없는데 1\*1 필터를 사용하는 이유? 5\*5 에 패딩을 입힐 이유가 없지않나

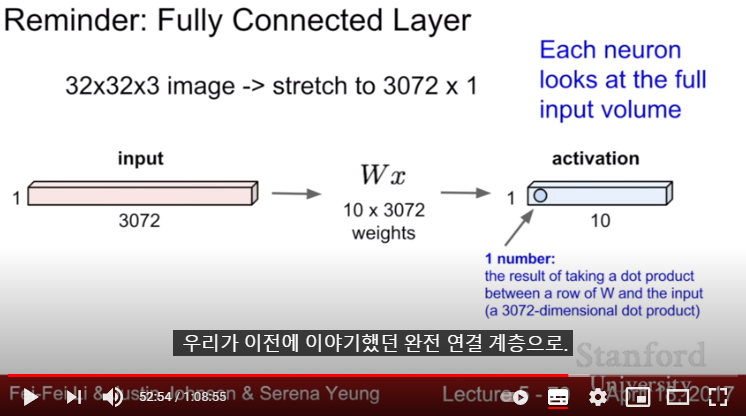


어제까진 이상하게 한글 번역이 안됏는데 갑자기 돼서 좀 진도가 빨라짐

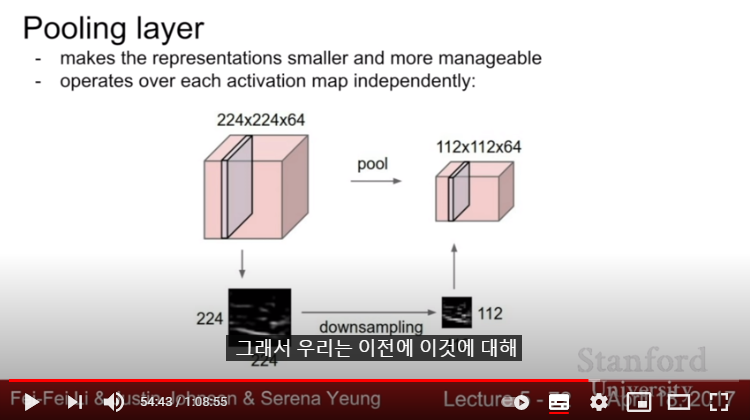
하나의 필터에서 나온 한개의 숫자는 실제 뇌와 뉴런의 connection과 유사한 구조라고 함

필터의 수가 많아지면 많아진 만큼 서로 다른 특징의 레이어가 생김

영역이 같아도 결국 필터ㅊㅇ가 생김



이건 앞에서(강의 초반) 봤던 Fully connected layer를 방금 배운 내용으로 설명하는 내용



이제 아까 궁금했던 Pool이 무엇인지 설명

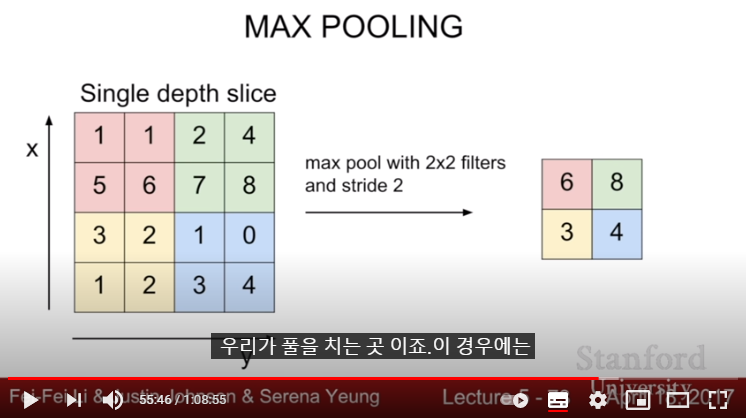
풀링 레이어가 하는 일은 일종의 다운 샘플링임.

앞서 레이어가 작아지는걸 막기위해 사용한게 Padding이라면,

Pooling은 사이즈를 줄이는 역할을 함(굳이 써야하나? - > 나중에 질문이 나왔고, 요즘은 pooling을 주기보다는 그냥 stride를 늘리는 식을 선호한다고함)

다만 그냥 줄이는게 아니라 레이어의 핵심적인 특징은 유지하면서 줄이는데,

그 방법은 그냥 작은걸 버리는것임



이렇게ㅇㅇ

4\*4 size에서 2\*2 필터에 stride를 2를 주면 위와 같이 4등분하여 pooling을 하게 됨

여기까지가 5강의 내용...

