

CNN

사각피질

- 허블Hubel 과 비셀Wiesel 의 시각 체계 연구(고양이 실험)
 - 뉴런들은 일부 시야 범위 내의 시각 자극에만 반응
(국부 수용장을 local receptive field 가짐)
 - 단순한 형태에 자극 받는 뉴런이 있는 반면, 어떤 뉴런은 저수준 패턴이 조합된 더 복잡한 패턴에 반응
- 고수준 뉴런이 이웃한 저수준 뉴런의 출력에 기반하는 구나!

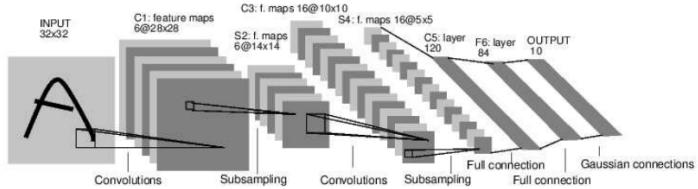


- Convolutional Neural Network (CNN)으로 진화
 - LeNet-5[LeCun98]
 - 수표의 손으로 쓴 숫자 인식
 - 합성곱층Convolution layer, 풀링층pooling layer 제안



Yann LeCun

Facebook AI & New York Univ.



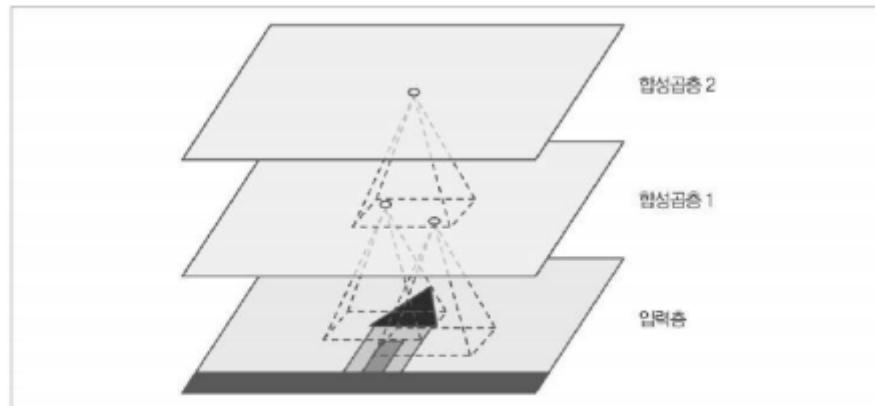
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

합성곱층

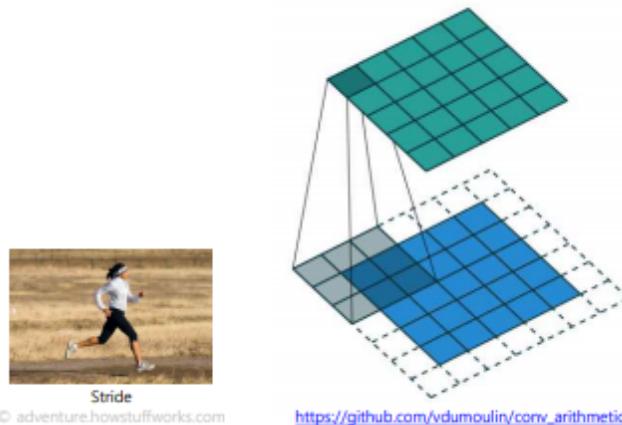
CNN 구조

- CNN 의 주요 구성요소: 합성곱층 convolutional layer
 - 수용장 안에 있는 영역만 상위 레이어 유닛에 연결
 - 첫 번째 히든 레이어는 저수준 특성에 집중, 두 번째 히든 레이어는 고수준 특성으로 조합해 나감

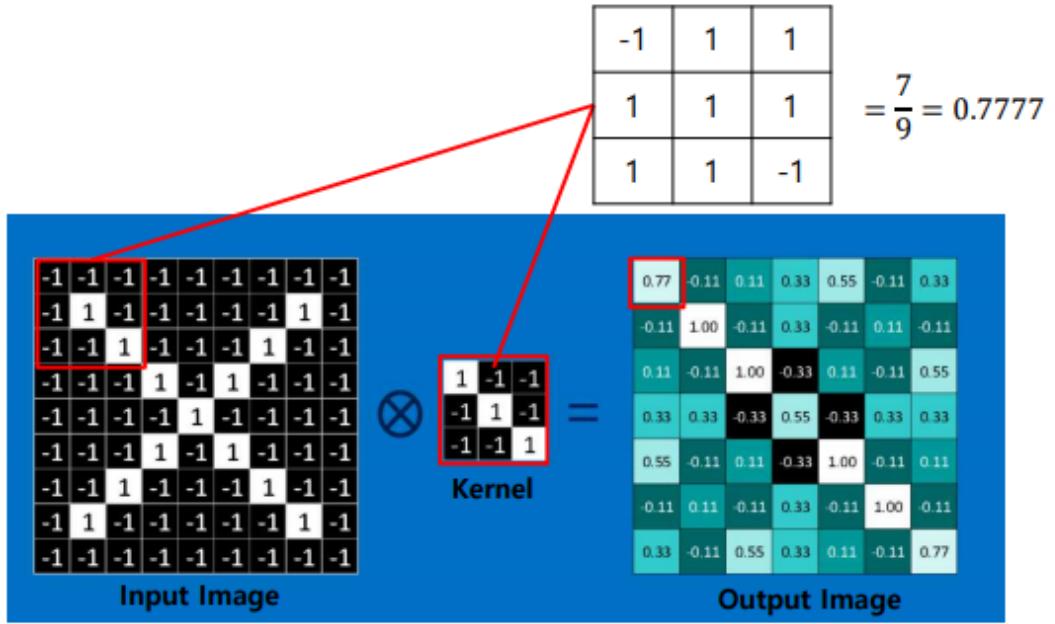
그림 13-2 사각 형태의 국부 수용장을 가진 CNN 층



- 한 함수 g 가 다른 함수에 대해 얼마나 겹치는 지 계산
 - Kernel(or filter) : 함수 g
 - Stride : 커널의 이동 보폭
 - Padding : 출력 크기를 조정하기 위해 입력 가장자리에 셀을 추가

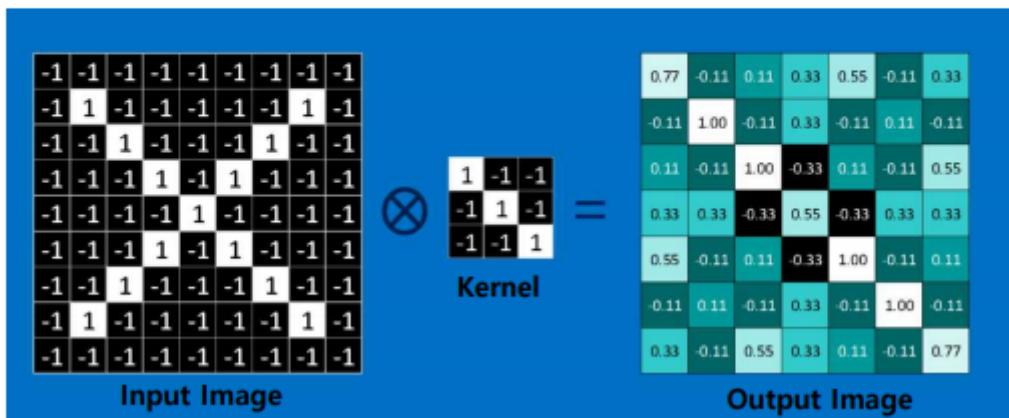


합성곱연산



https://brohrer.github.io/how_convolutional_neural_networks_work.html

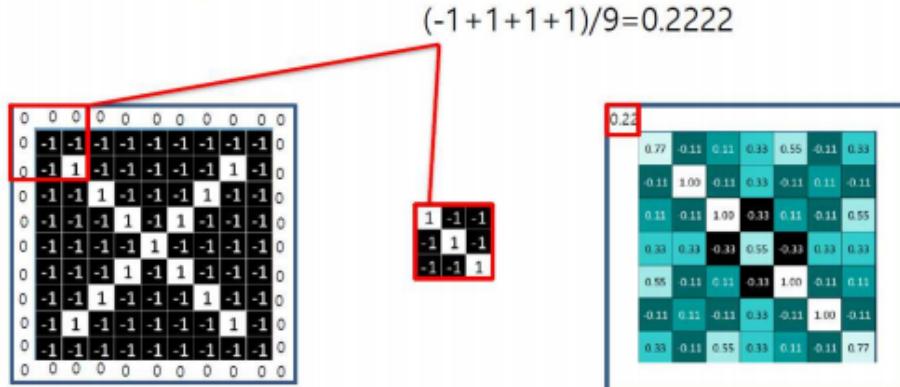
- 출력 영상 Output Image 의 크기: $\text{floor}((I - F)/S + 1)$
 - I: 입력 영상 Input image 의 크기
 - F: 커널의 크기
 - S: 스트라이드



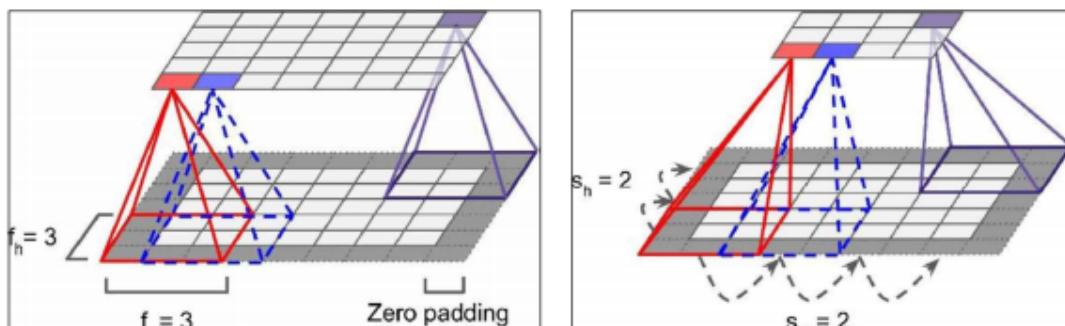
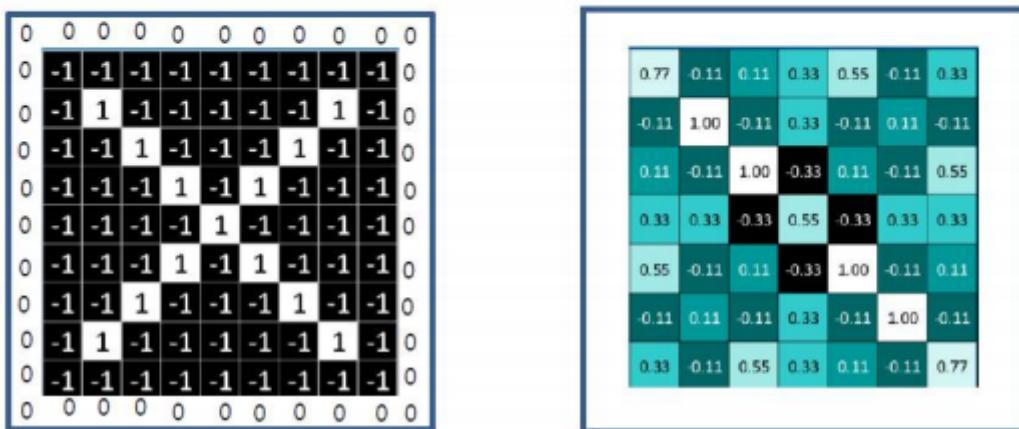
커널을 n번 적용할 경우 Output의 크기

$> 9 * 9 \rightarrow 5 * 5 \rightarrow 3 * 3 \rightarrow 1 * 1$

- 패딩 Padding: 가장 자리에 공간을 추가하여 크기를 조절
 - 제로 패딩 Zero padding: 추가된 공간을 0으로 채움



- 출력 영상의 크기: $\text{floor}((I - F + 2P)/S + 1)$
 - I: 입력 영상 Input image 의 크기
 - F: 커널의 크기
 - S: 스트라이드
 - P: 패딩으로 추가된 공간의 크기



레이어 사이의 연결 모습과 제로 패딩

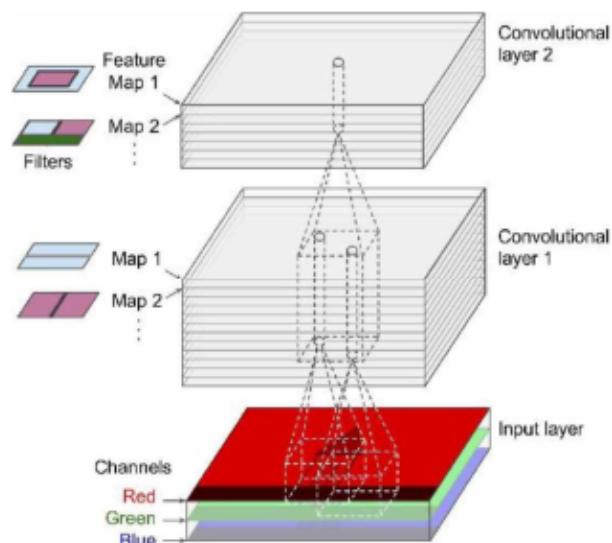
스트라이드로 특성맵 크기 축소시키기

- 고전적인 필터들은 사람들이 직접 고안해 내었다.
 - ex) blur 처리를 하려면 gaussian filter를 연구, outline 필터를 만드려면 sober필터를 연구 등등
- 최근에는 필터들을 학습시킴. (Automatically learned filter)

합성곱식

- 합성곱 레이어에 있는 유닛의 출력 계산

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_n'-1} x_{i',j',k'} \times w_{u,v,k',k}, \text{ where } \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$



- 식을 보면 계속 공부했다 싶이 가중합과 편향을 더하는 꼴이란 것을 확인할 수 있음.
- l = 합성곱층, k = 특성 맵, i = 특성맵의 행, j = 특성맵의 열
- 합성곱 층 l 에 있는 k 특성 맵의 i 행, j 열에 위치한 뉴런은 이전 $l-1$ 층에 있는 모든 특성 맵의 $i * s_h$ 에서 $j * s_h + f_h - 1$ 까지의 행과 $i * s_w$ 에서 $j * s_w + f_w - 1$ 까지의 열에 있는 뉴런의 출력에 연결된다.
- 다른 특성 맵이더라도 같은 i 행과 j 열에 있는 뉴런이라면 정확히 이전 층에 있는 동일한 뉴런들의 출력에 연결됨

tesorflow keras - Conv2d

```
conv = keras.layers.Conv2D(filters = 32, kernel_size = 3, strides = 1, padding = "same", activation = "relu")
```

- 필터의 수, 커널의 크기, 스트라이드 값, 패딩 값, 활성화 함수 등등
- 이 convolution layer 파라미터의 수는?

Same vs Valid 패딩

- Same: 제로 패딩이라고 하며, (출력의 사이즈 = 입력의 사이즈)를 맞춰주기 위해서 자동적으로 0으로 패딩을 넣어줌

간혹 stride 크기로 인해서 same패딩이라고 지정을 해주었지만 출력 사이즈가 다를 수 있음

- Valid: 패딩 없음

###

풀링층

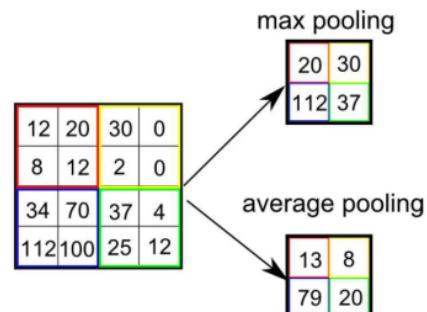
- 풀링 pooling(부표본 subsampling)

- 정보의 압축, 축소
- 중요한 정보 또는 평균 정보만 전달
- 보통 입력 채널 수 = 출력 채널 수



- 종류

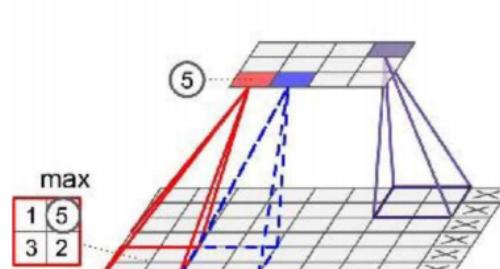
- 최대 풀링 Max pooling
 - 강한 자극만 기억
- 평균 풀링 Average pooling
 - 평균 자극을 기억



- 보통 cnn에서 이미지 특성 크기를 줄이는데 사용

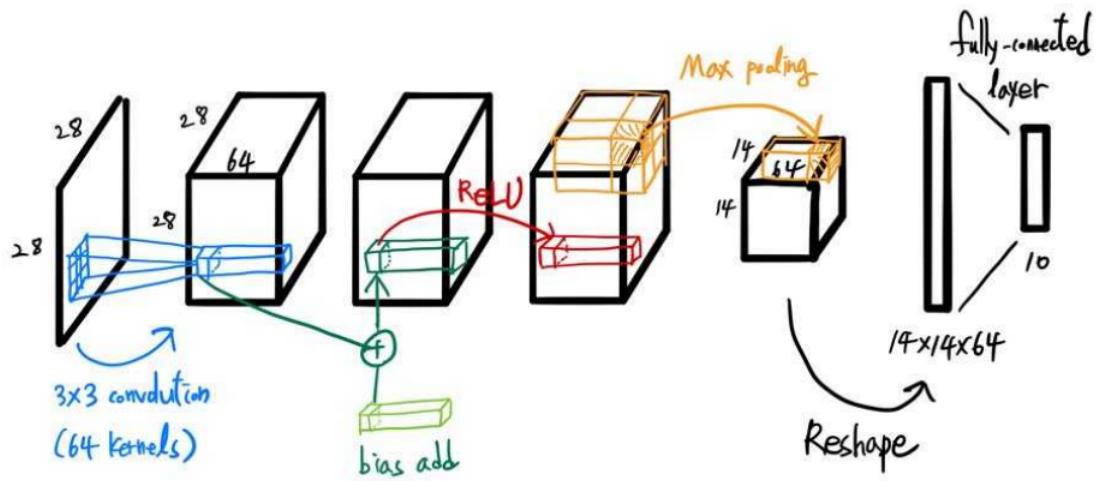
- 목적

- 계산량과 메모리 사용량 ↓
- 파라미터 수 ↓(과적합 방지)
- 위치 불변성 location invariance

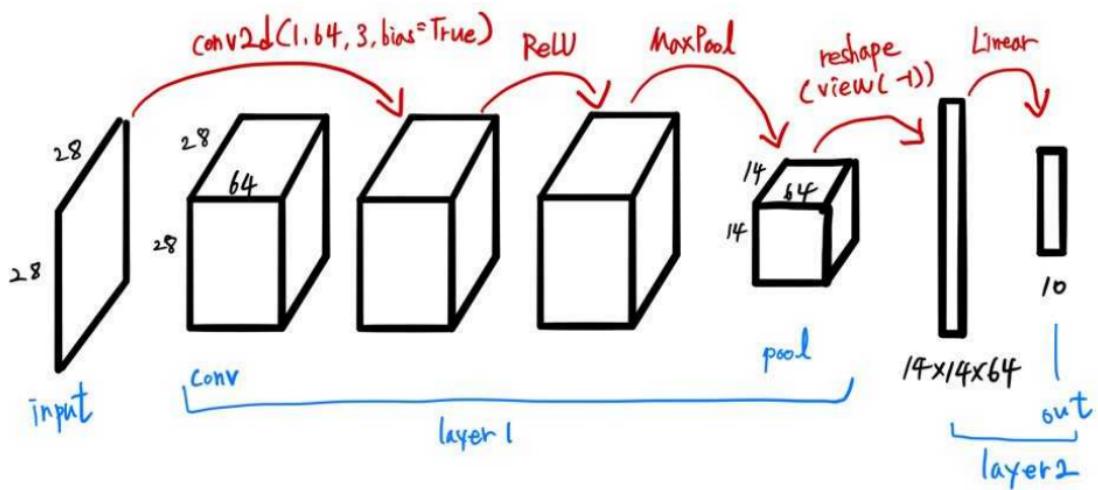


최대 풀링층(2x2 풀링 커널, 스트라이드2, 패딩 없음)

CNN 기본 구조

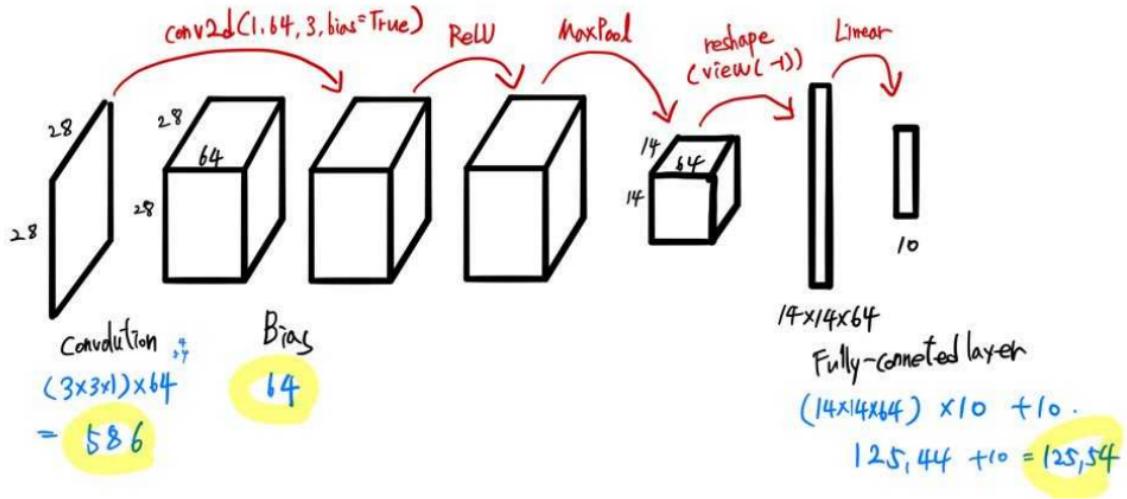


- 기본 구조: 합성 곱층, 풀링층



- 합성곱층 + 풀링층 하나씩을 layer 하나라고 한다. 후에 fully connected (다층 퍼셉트론) 부분을 다른 한 layer로 본다.

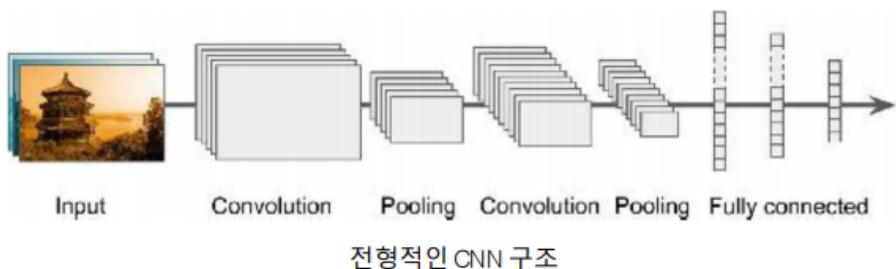
합성곱층 + 풀링층!!! (한 세트)



- parameter 계산 방법...

고급 CNN

- 전형적인 CNN 구조
 - 몇 개의 합성곱 레이어 (각각 ReLU 레이어)
 - Pooling 레이어
 - ReLU 레이어
- 이미지는 작아지고, 깊어짐(채널 수 증가)
- 구현 팁: 5x5 커널 하나 쓰기 vs. 3x3 커널 두 개 쓰기
 - 3x3 커널 두 개를 이어서 5x5 커널과 같은 효과 낼 수 있음
 - 파라미터 계산량이 훨씬 적음
(예: 3x3 두 개 → 18개, 5x5 한 개 → 25개)



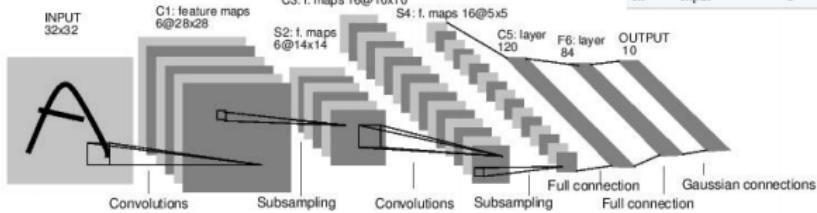
- CNN구조에서는 ILSVRC ImageNet 대회 분류 작업 top-5에 러율을 차지한 영향력 있는 "AlexNet, GoogleNet, ResNet을 위주로,,," + (cnn의 근본 LeNet-5) 도 같이 공부
- 개인적으로 miniVGGNet도 혼자서 공부하는게 좋을 듯

LeNet-5

- CNN 의 고전으로 알려짐
- Yann LeCun, MNIST 에 사용됨
- 28×28 픽셀에 제로 패딩 $\rightarrow 32 \times 32$
- 요즘엔
 - Max pooling 선호됨
 - 5×5 커널 한 개 $\rightarrow 3 \times 3$ 커널 두 개
 - RBF \rightarrow Softmax

Table 13-1. LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–



(데모) <http://yann.lecun.com/exdb/lenet/index.html>

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

AlexNet

- Top-5 에러율: 17% (2위 26%)
- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
- 규제 기법
 - F8 과 F9 출력에 드랍아웃(Drop out)(50%)
 - 데이터 증식(Data augmentation)
 - : 랜덤하게 이동, 수평 뒤집기, 밝기 변경 등

Table 13-2. AlexNet architecture

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	–
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	–
C1	Convolution	96	55×55	11×11	4	SAME	ReLU
In	Input	3 (RGB)	224×224	–	–	–	–

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012. p. 1097-1105.

AlexNet에서 주요하게 볼 것

• Local response normalization (LRN)

$$- b_i = a_i \left(k + \alpha \sum_{j=j_{bw}}^{j_{high}} a_j^2 \right)^{-\beta} \quad w \text{if } \begin{cases} j_{high} = \min(i + \frac{r}{2}, f_n - 1) \\ j_{bw} = \max(0, i - \frac{r}{2}) \end{cases}$$

- b_i : u 행, v 열에 위치한 뉴런의 정규화된 출력
- a_i : ReLU 단계 후 정규화 거치기 전의 뉴런의 활성화 값
- k, α, β, r : 하이퍼파라미터(k : bias, r : depth radius)

- 역할

- 강하게 활성화된 뉴런이 다른 특성 맵의 동일 위치의 뉴런을 억제
(생물학적 뉴런의에서 영감-경쟁적인 활성화 competitive activation)
- 더 넓은 시야에서 특징을 탐색 → 일반화 성능 향상

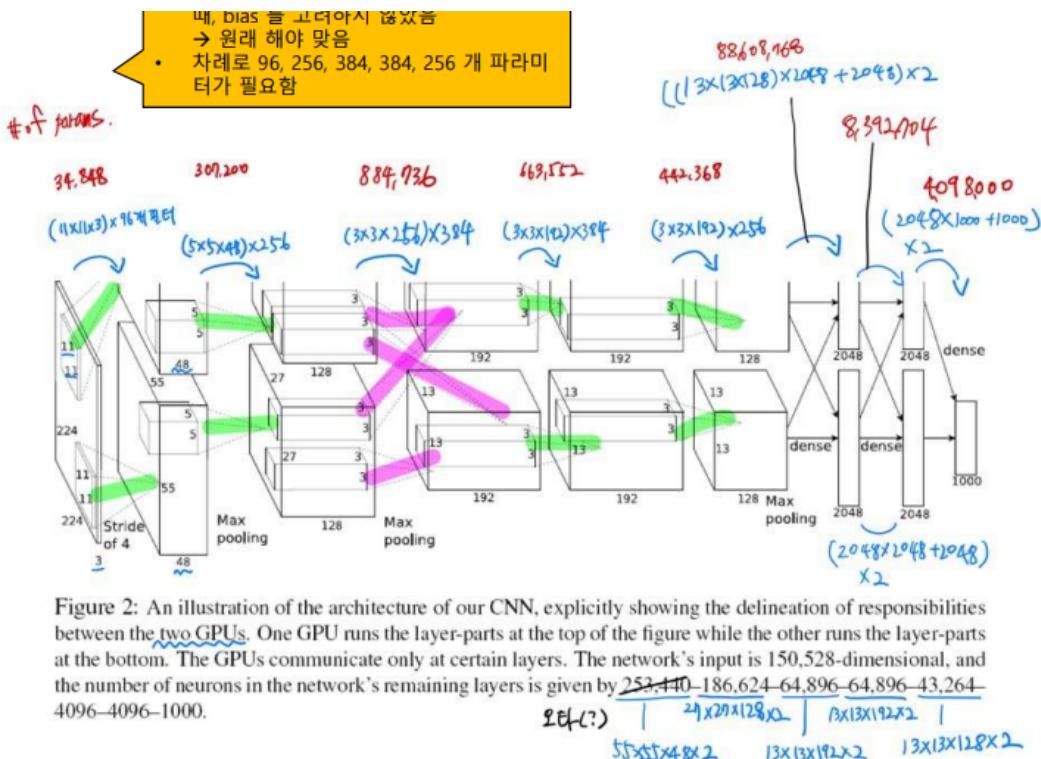


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by $\frac{253,440 - 186,624 - 64,896 - 64,896 - 43,264 - 4096 - 4096 - 1000}{55 \times 55 \times 48 \times 2} = 29 \times 27 \times 128 \times 2$.

GoogleLeNet

- Top-5 에러율 7% 이하
- 이전 CNN 보다 훨씬 깊은 구조(22 레이어)

- 인셉션 모듈 Inception module
 - 효과적으로 파라미터 사용
 - 파라미터 수: AlexNet 의 1/10 ($60M \rightarrow 6M$)
 - 예) $3 \times 3 + 1(S)$
 3×3 커널, 스트라이드 1,
 SAME 패딩
 (입/출력의 너비, 높이 같음)
 - `torch.cat()`
`tf.concat()`

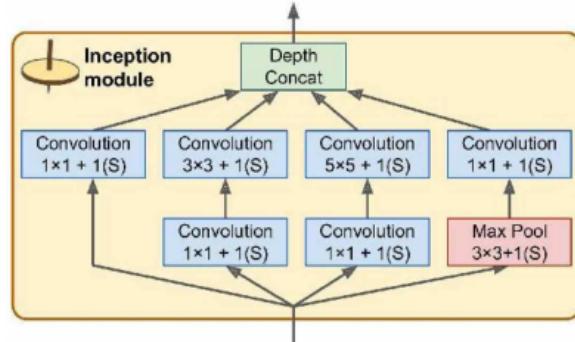


Figure 13-10. Inception module

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

- 인셉션 모듈의 큰 특징은 같은 크기, 사이즈의 출력들을 깊게 이어 붙힌다!
- "SAME" 패딩으로 2번째 layer들의 필터 사이즈가 다 달라도 출력의 사이즈는 다 같으니 깊게 이어 붙힐 수 있음.

• 인셉션 모듈 Inception module

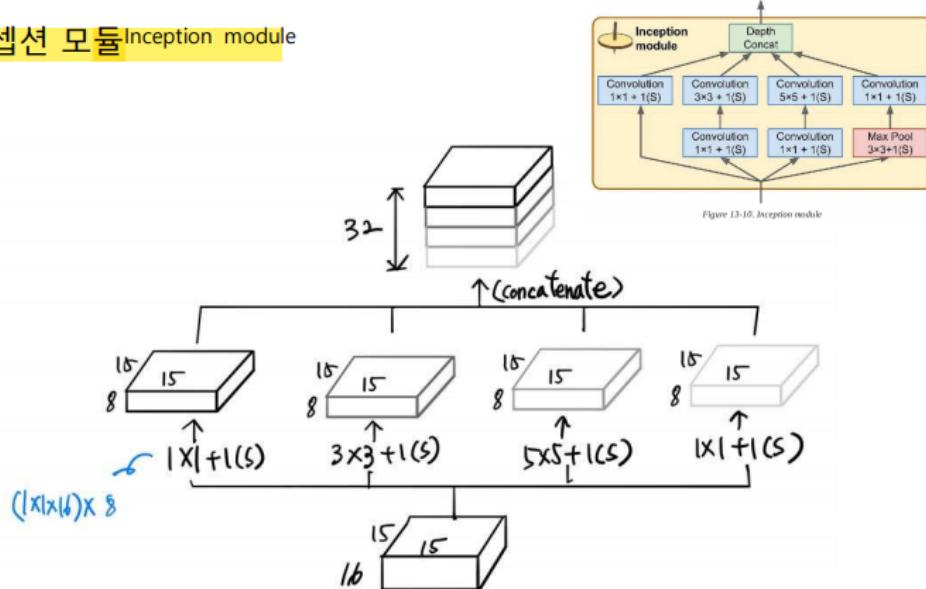
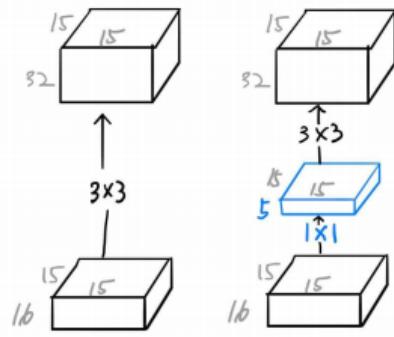


Figure 13-10. Inception module

- 1x1 커널을 왜 사용? 1x1 커널을 생각해보면 한 픽셀씩 읽어들여서 무슨 특징을 잡을 수 있는지에 대한 궁금증.....

- 1x1 커널 컨볼루션 레이어 역할
 - 차원 축소
 - 입력 레이어보다 더 적은 특성 맵 Feature map 을 출력
→ 차원 축소
 - 병목층 Bottleneck layer
 - 3x3 과 5x5 컨볼루션 전에 유용
 - 더 복잡한 패턴을 감지할 수 있는 한 개의 강력한 컨볼루션 레이어처럼 작동
 - [1x1, 3x3]과 [1x1, 5x5]



- 직관적 이해: 복잡한 패턴을 갖는 여러 크기의 특성 맵을 출력하는 각성제를 맞은 컨볼루션 레이어라 생각
- 컨볼루션 커널의 수는 하이퍼 파라미터

- 위의 그림을 보면 왼쪽에 인셉션 모듈을 사용하지 않게 될 때의 파라미터 개수는 $(3 \times 3 \times 16) \times 32 = 4608$
- 오른쪽의 파라미터 개수는 $(1 \times 1 \times 16 \times 5) + (3 \times 3 \times 5 \times 32) = 1520$
적은 파라미터의 개수로 더 강력한 패턴을 갖는 특성맵을 얻을 수 있음

• 전체 구조

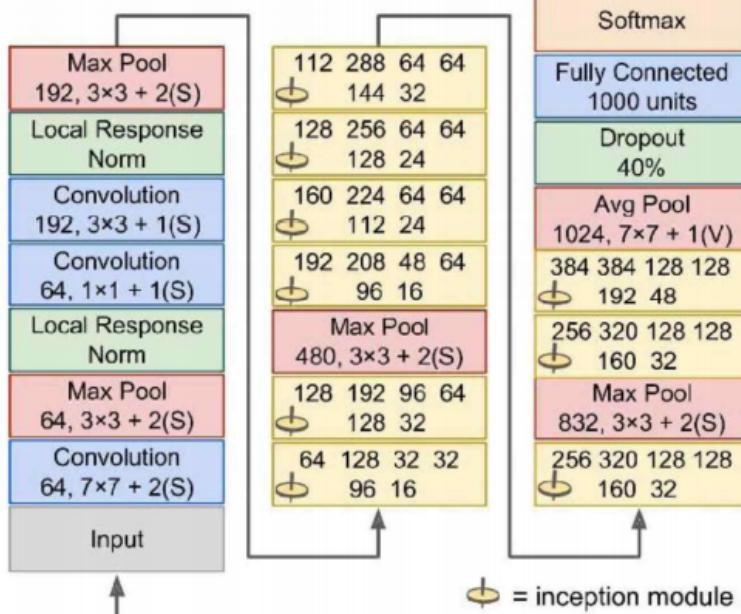
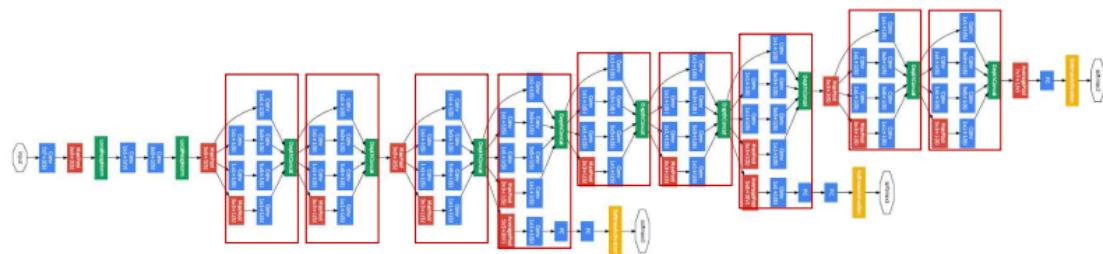


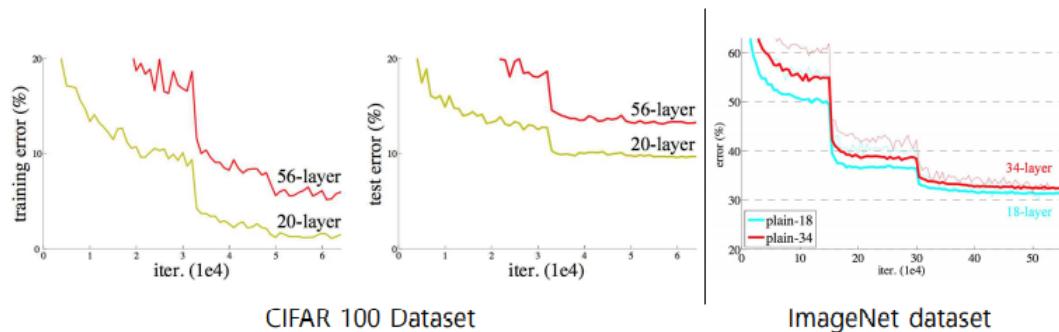
Figure 13-11. GoogLeNet architecture

- 전체 구조
 - Network in network!
 - Inception module: subnetwork



ResNet

- 레이어들을 깊게 만들고 싶다는 생각
 - GoogLeNet 이후 생각들
 - 깊은 네트워크가 항상 좋은가?



- 스킵 커넥션 Skip connection(숏컷 커넥션 Shortcut connection)

- 어떤 레이어에 입력되는 신호가 상위 레이어의 출력에도 더해짐
- (좌) 일반적인 뉴럴 네트워크
 - 초기화 가중치가 0에 가깝기 때문에 출력 또한 0에 가까움
- (우) 스킵 커넥션을 가진 뉴럴 네트워크
 - 입력과 같은 값을 출력

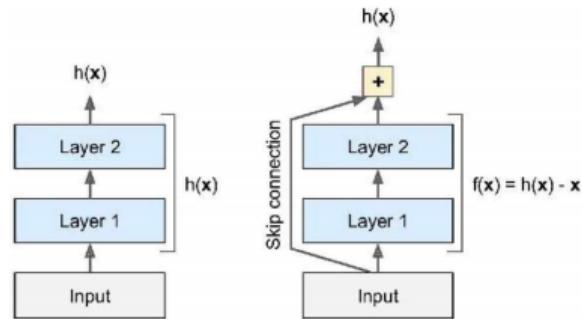


Figure 13-12. Residual learning

- 스킵 커넥션 Skip connection(숏컷 커넥션 Shortcut connection)

- 스킵 커넥션을 많이 추가할 경우, 일부 레이어가 미학습 상태더라도 훈련 시작 가능 (gradient vanishing 문제 방지)
- 잔차 유닛 Residual unit

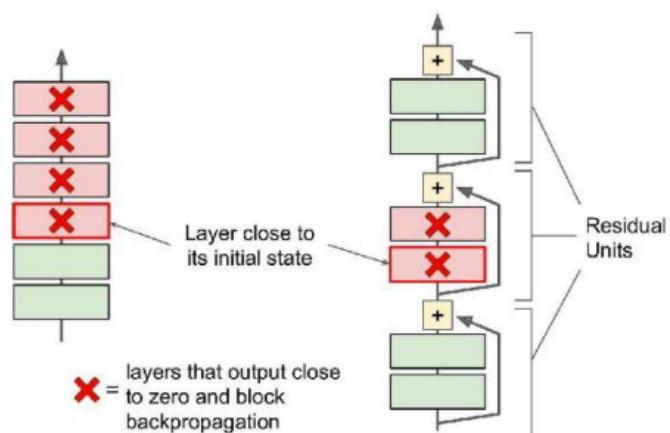


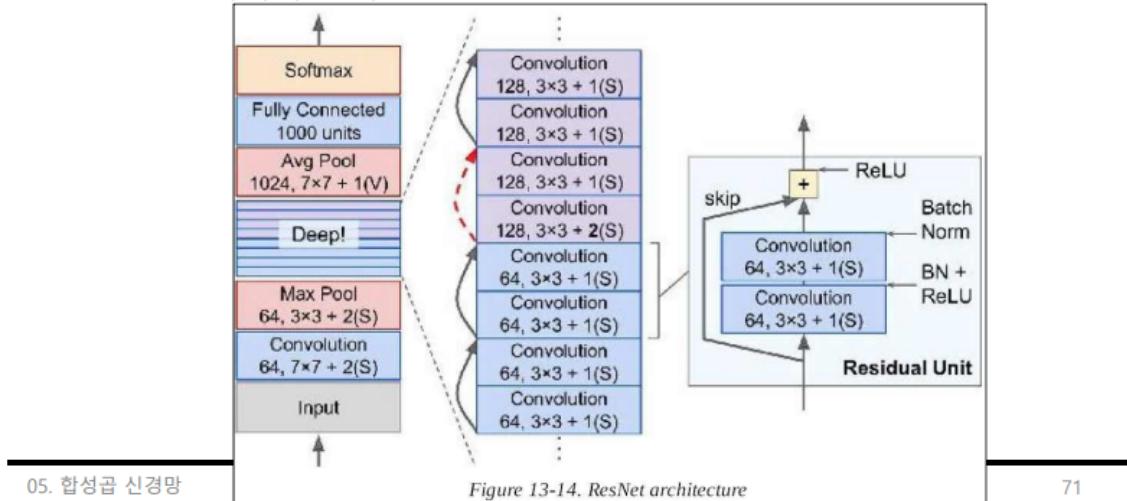
Figure 13-13. Regular deep neural network (left) and deep residual network (right)

- 왼쪽의 그림은 output에 가까울 수록 gradient가 0의 값에 가깝게 변하게 됨 --> backpropagation 을 진행할 수 없게된다.
- 깊은 레이어에서는 gradient 가 0이 되는 일이 잦음--> gradient 소실 문제
- 즉 skip connection을 사용하여 gradient가 0과 가까우면 그 unit을 backpropagation에서 건너뜀

- 전체 구조

- 몇 개 residual unit 마다

- 특성 맵 Feature map 의 수 → 2배씩 늘어남
- 너비와 높이 → 절반



- 전체 구조

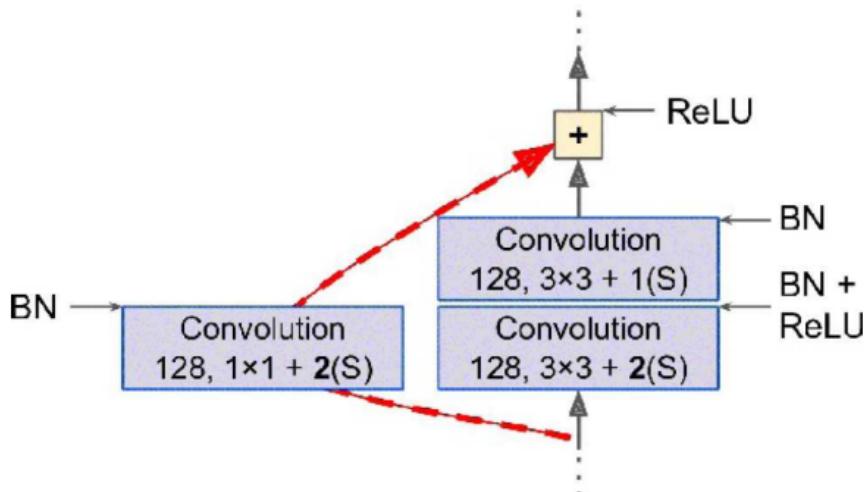


Figure 13-15. Skip connection when changing feature map size and depth

• 전체 구조

