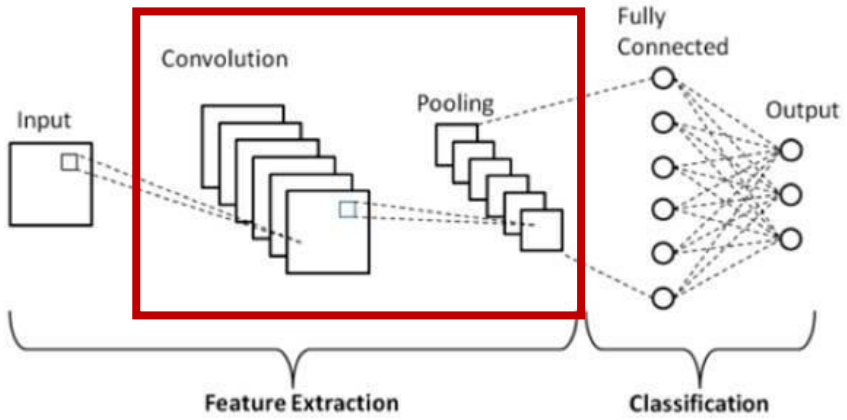
전형적인 CNN의 구조

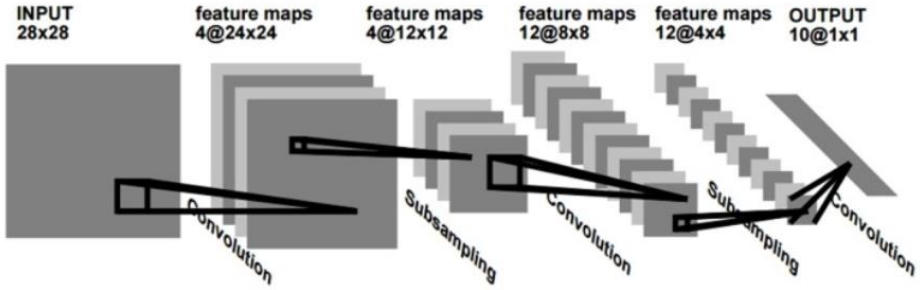


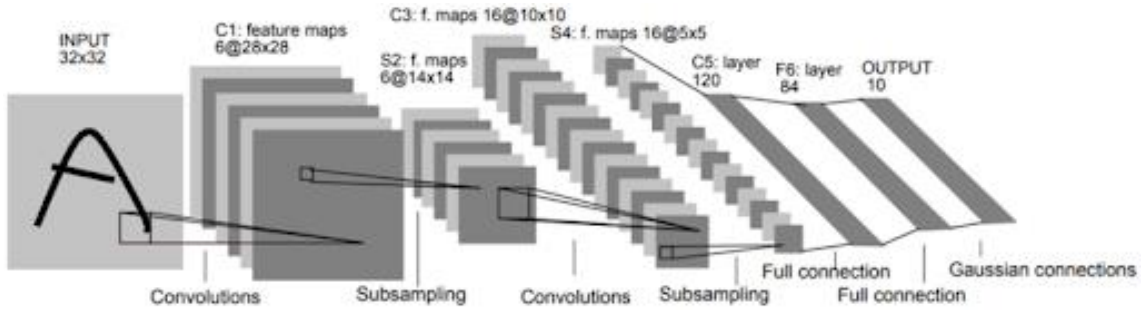

```
3   model = keras.models.Sequential([
4       Conv2D(filters=64, kernel_size=7, activation='relu', padding='same',
        input_shape=[28, 28, 1]),
5       keras.layers.MaxPooling2D(2),
6       Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'),
7       Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'),
8       keras.layers.MaxPooling2D(2),
9       Conv2D(filters=256, kernel_size=3, activation='relu', padding='same'),
10      Conv2D(filters=256, kernel_size=3, activation='relu', padding='same'),
11      keras.layers.MaxPooling2D(2),
12      keras.layers.Flatten(),
13      keras.layers.Dense(128, activation='relu'),
14      keras.layers.Dropout(0.5),
15      keras.layers.Dense(64, activation='relu'),
16      keras.layers.Dropout(0.5),
17      keras.layers.Dense(10, activation='softmax'),
18  ])
```

Convolution Layer – Pooling Layer - ⋯ – Fully-connected Layer

Best CNN architecture - LeNet5

· LeNet의 발전





LeNet-1 (1990) → **LeNet-5 (1998)**

- CNN을 최초 개발한 Yann LeCun이 우편번호의 필기체를 인식하기 위한 용도로 개발 시작.
- 1998년 최종적으로 발표한 구조가 LeNet5
- 초기에는 컴퓨팅 능력이 떨어져 파라미터 수가 작은 망을 개발.
- LeNet5에서는 성능 향상을 위 해 더 큰 구조로 설계.

Best CNN architecture - LeNet5

· LeNet의 구조

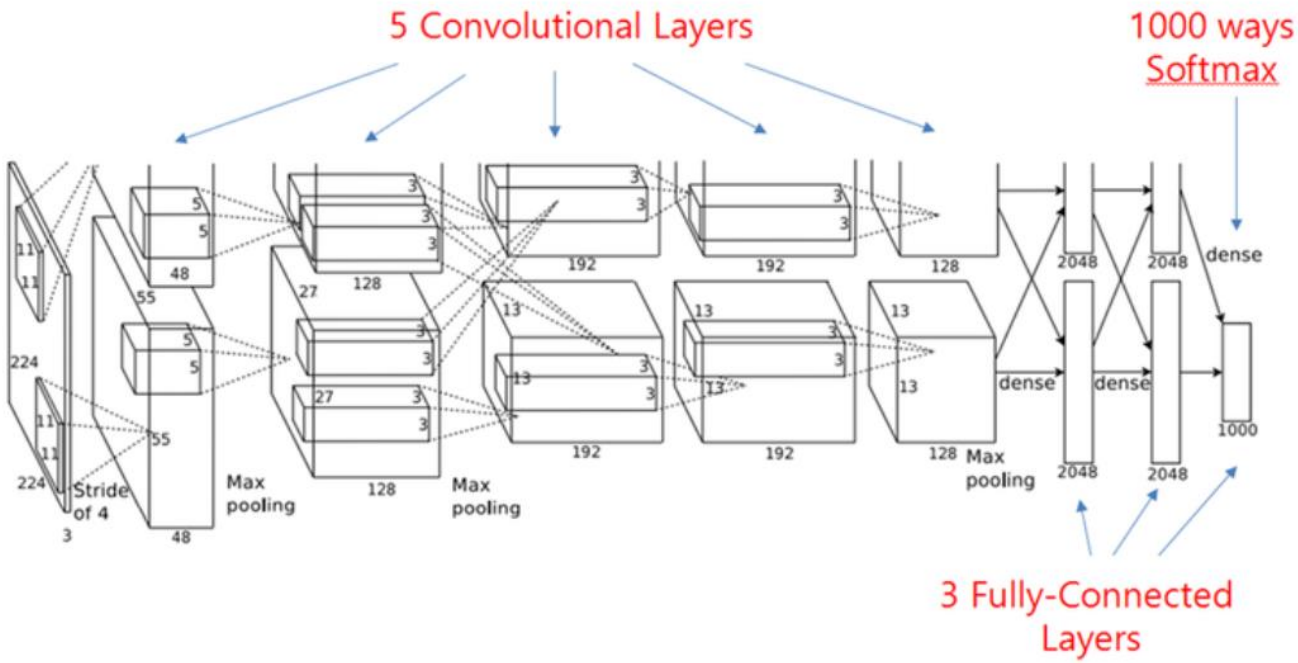
Convolution Layer : 3
Pooling Layer : 2
Fully-connected Layer :2

Layer	Layer Type	Feature Maps	Kernel(Filter)	Input size	Trainable para	Activation
Input	image	-	-	32x32	-	-
C1	Conv	6	5x5	28x28	156	Sigmoid
S2	Sub Sampling	6	2x2	14x14	12	Tanh
C3	Conv	16	5x5	10x10	1516	Tanh
S4	Sub Sampling	16	2x2	5x5	32	Sigmoid
C5	Conv	120	5x5	1x1	48120	Tanh
F6	Dense	-	-	84	10164	Tanh
Output	Dense	-	-	10	-	Softmax

```
1
2
3   lenet_5_model = keras.models.Sequential([
4       keras.layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh',
        input_shape=train_x[0].shape, padding='same'), #C1
5       keras.layers.AveragePooling2D(), #S2
6       keras.layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh',
        padding='valid'), #C3
7       keras.layers.AveragePooling2D(), #S4
8       keras.layers.Flatten(), #Flatten
9       keras.layers.Dense(120, activation='tanh'), #C5
10      keras.layers.Dense(84, activation='tanh'), #F6
11      keras.layers.Dense(10, activation='softmax') #Output layer
12  ])
13
```

- Input을 32*32로 받아 영상의 중앙에 위치시켜 처리 (이미지의 디테일에 대한 고려가 많아져 더 우수한 성능을 보임)
- Padding을 same으로 주어 특징맵의 사이즈가 input과 동일한 size로 출력하게 함.

Best CNN architecture - AlexNet



- ImageNet 영상 데이터베이스 기반 대회 ILSVRC(2012)에서 우승한 팀이 개발한 CNN 구조.
- 구조적 관점에서 LeNet5와 크게 다르지 않지만 높은 성능을 얻기 위해 다양한 고려가 들어감.
- 특히 GPU를 사용해 의미 있는 결과를 얻어 이후 CNN구조 설계시 GPU 사용이 대세가 됨.
- 6억 3000만개의 연결로 구성된 방대한 CNN 구조.

Best CNN architecture - AlexNet

Convolution Layer : 5
Pooling Layer : 3
Fully-connected Layer :3

Type / Stride / Padding	Filter Shape	Input Size
Conv / s4 / p2	11 x 11 x 3 x 96	224 x 224 x 3
Conv / s1 / p2	5 x 5 x 96 x 256	55 x 55 x 96
Max Pool / s2	3 x 3	-
Conv / s1/ p1	3 x 3 x 256 x 384	27 x 27 x 256
Max Pool / s2	3 x 3	-
Conv / s1 / p1	3 x 3 x 384 x 384	13 x 13 x 384
Conv / s1 / p1	3 x 3 x 384 x 256	13 x 13 x 384
Max Pool / s2	3 x 3	-
FC	9216 x 4096	9216
FC	4096 x 4096	4096
FC	4096 x 1000	4096

1. 활성화함수 ReLU 사용

2. Multiple GPUs

: 2개의 GPU를 사용한 병렬적 구조 → one-GPU 네트워크에 비해 error rate을 약 1.7% 낮춰줌.

3. Local Response Normalization(LRN)

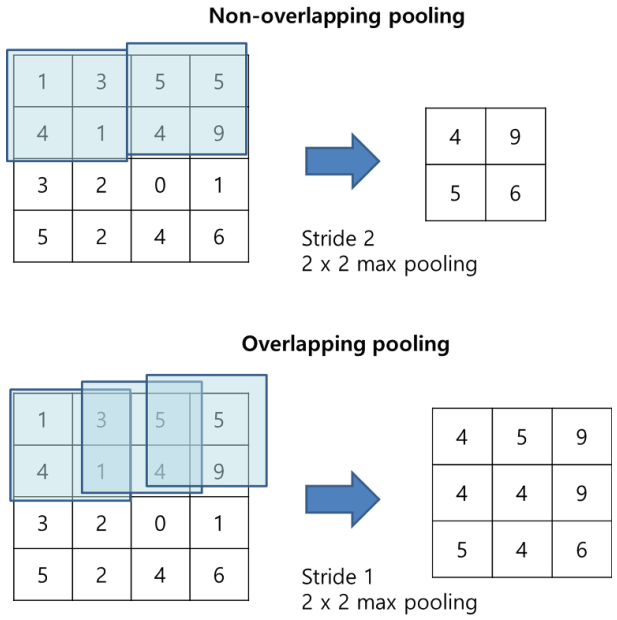
$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2)^\beta$$

where
$b^i_{x,y}$ – regularized output for kernel i at position x, y
$a^i_{x,y}$ – source ouput of kernel i applied at position x, y
N – total number of kernels
n – size of the normalization neigbourhood
$\alpha, \beta, k, (n)$ – hyperparameters

- Convolution Layer 다음 Local Response Normalization 수행.
- 같은 위치의 픽셀에 대하여 복수의 feature map 간 정규화를 하는 기법.
- ReLU를 사용하게 되면 양수의 방향으로 입력 값을 그대로 사용 → Pooling시 매우 큰 하나의 픽셀 값이 주변의 픽셀에 영향을 미치게 됨 → 이를 방지하기 위해 같은 위치에 있는 픽셀끼리 정규화.

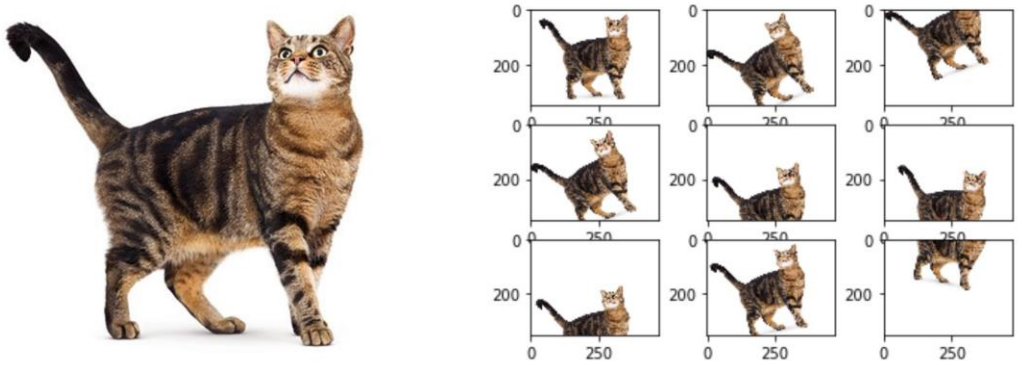
Best CNN architecture - AlexNet

4. Overlapped Pooling



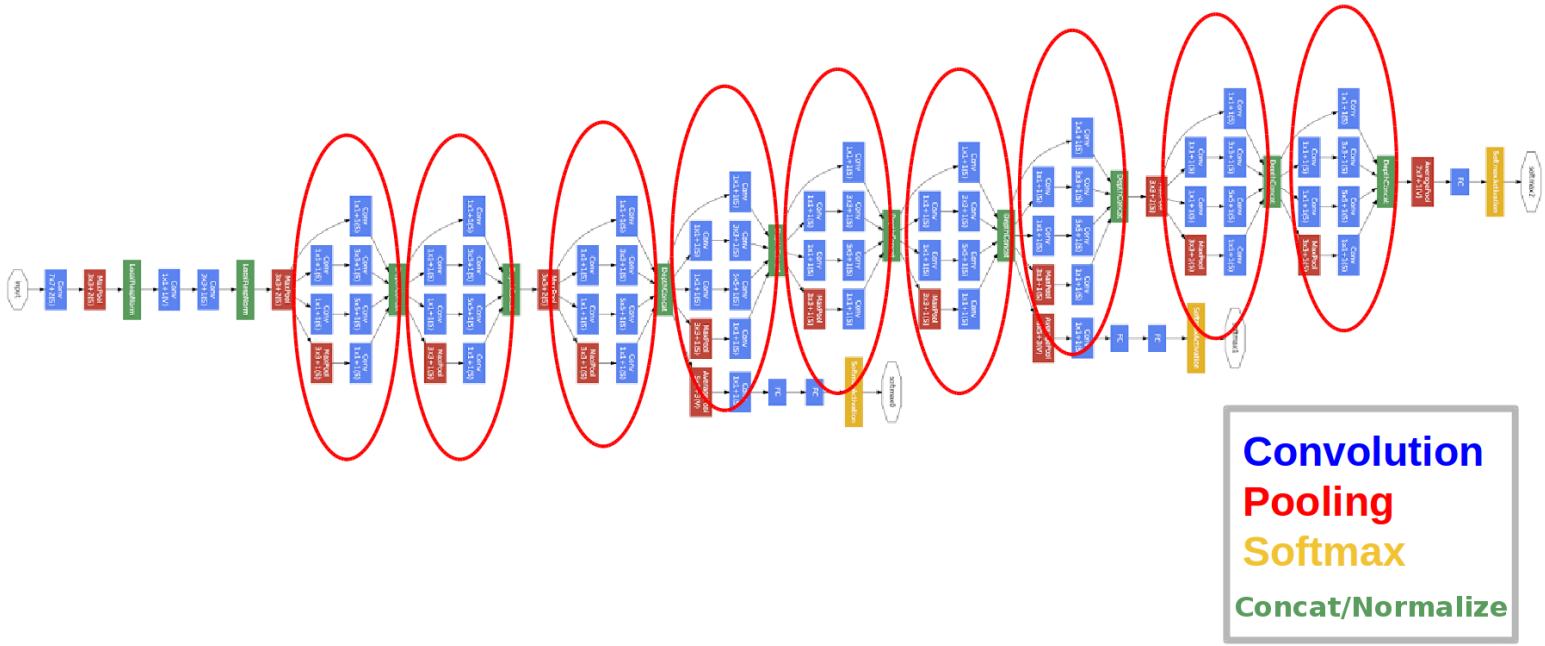
- Pooling은 일반적으로 필터를 겹치지 않게 Stride를 적절히 조정하여 사용.
- AlexNet에서는 Stride를 좁혀 Overlapping 하는 구조 사용.
- 정확도는 약 0.4%가 향상되나, Overlapping의 사용은 연산량을 증가시킴.

5. Data Augmentation(데이터 증식)



- 동일한 이미지들을 조금씩 변형시켜가며 학습 Overfitting을 방지
- Mirroring / Random Crops / PCA Color Augmentation 등
- AlexNet에서는 데이터 양을 약 2048배 증진.

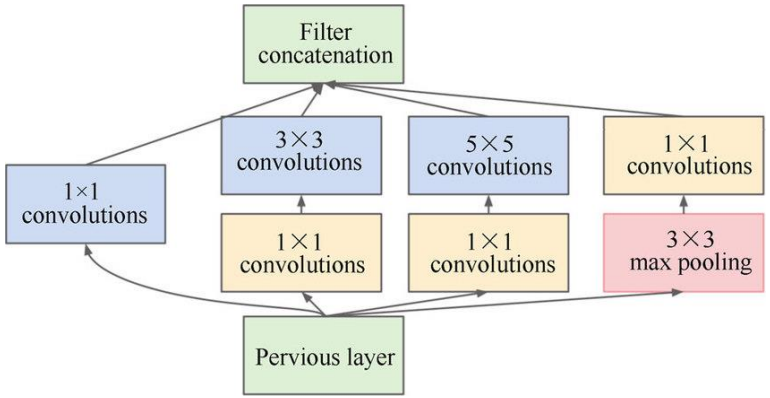
Best CNN architecture - GoogLeNet



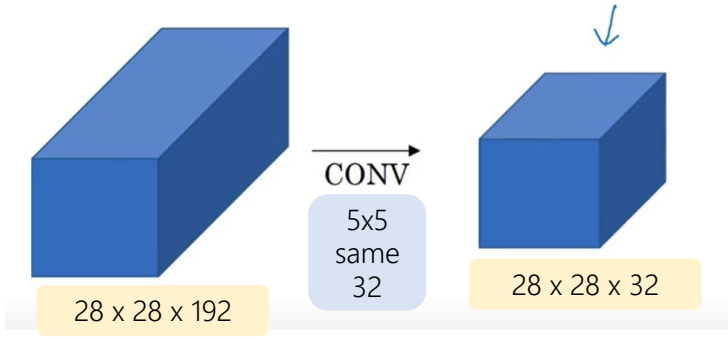
- ILSVRC(2014)에서 우승을 차지한 구조. 구글 리서치에서 개발.
- 인셉션 모듈(Inception Module)을 사용해 효과적으로 파라미터 사용
- 실제로 AlexNet보다 망의 깊이는 약 2배 이상 깊어졌으나, 10배 적은 파라미터를 가지고 에러율을 약 10% 낮추는 결과

Best CNN architecture - GoogLeNet

Inception Module

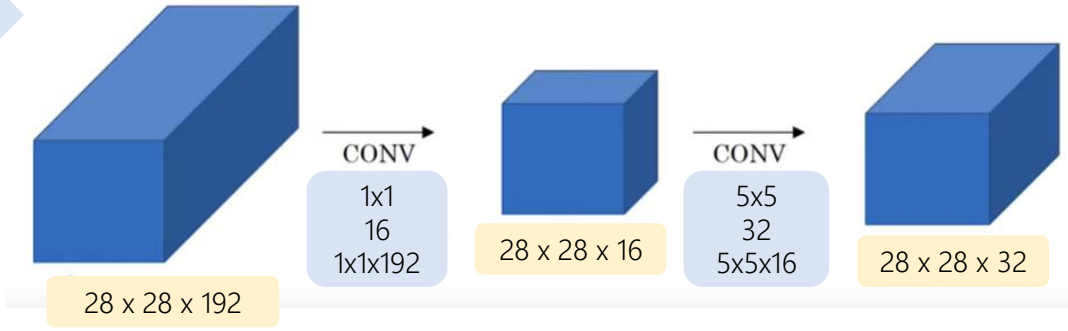


Using 1*1 Convolution Filter



28x28x32
x
5x5x192

= 120M



28x28x16x192
= 2.4M

+

28x28x32
x
5x5x16
=10M

= 12.4M

- 같은 Layer에 서로 다른 크기를 갖는 Convolution Filter를 적용하여 다른 scale의 feature을 얻을 수 있도록 함.
- 그림에서처럼, 1*1 filter를 적절히 사용하여 차원을 줄여 망이 깊어졌을 때 연산량이 늘어나는 문제를 해결.