합성곱 신경망(CNN)

2022.05

1. 영상 처리 기초

❖ 영상 처리(Image Processing)란?

- 입력 받은 영상을 사용 목적에 맞게 적절하게 처리하여 보다 개선된 영상을 생성하는 것
- 입력 영상에 있는 잡음(noise) 제거, 영상의 대비(contrast) 개선, 관심영역(region of interest) 강조, 영역 분할(segmentation), 압축 및 저장 등
- 저수준 영상 처리(좁은 의미의 영상 처리)
  - 영상 획득
  - 영상 향상
  - 영상 복원
  - 변환 처리
  - 영상 압축
- 고수준 영상 처리(컴퓨터 비전)
  - 영상 분할
  - 영상 표현
  - 영상 인식

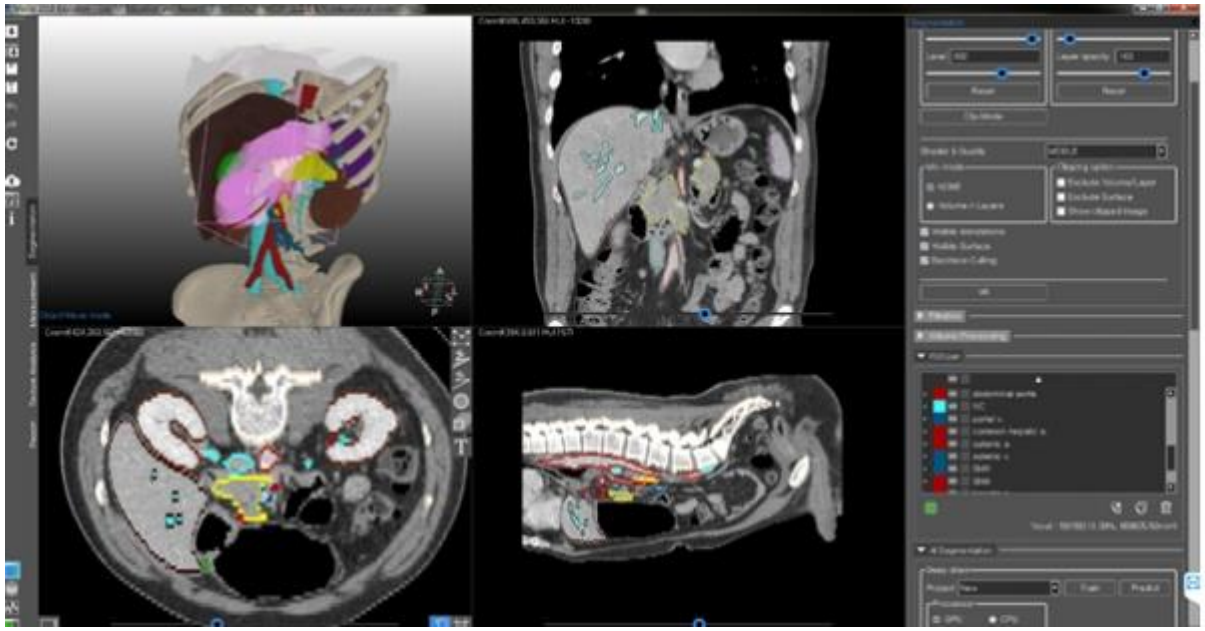
1. 영상 처리 기초

❖ 역사

- 영상 처리의 시작
  - 1920년대 초반 런던과 뉴욕 간에 해저 케이블을 통한 신문사들이 사진 전송
- 본격적인 영상 처리 위한 기술
  - 1940년대 폰 노이만의 디지털 컴퓨터의 개념 시작
  - 1950년 이후 트랜지스터, IC, 마이크로프로세서 같은 하드웨어 발달
  - 1950~60년대 프로그램의 언어의 발달과 운영체제 등의 소프트웨어 기술 발달
- 본격적인 영상 처리 시작
  - 우주 탐사 계획인 아폴로 계획과도 관련, 우주선에서 보낸 훼손된 영상의 복원 연구
- 1970년대 영상 처리 분야 더욱 발전
  - CT, MRI 등의 의료 분야
  - 원격 자원 탐사, 우주 항공 관련 분야
- 1990년대 컴퓨터 비전과 응용 분야 급속히 확장
  - 인터넷 시대에 영상검색, 영상전송, 영상광고
  - 디지털 방송 관련 컴퓨터 그래픽스, 디지털 카메라 보급

❖ 응용 분야

- 의료 분야 (방사선, 초음파)
  - 컴퓨터 단층촬영(CT), 자기 공명영상 (MRI)
  - 양전자 단층촬영(PET)

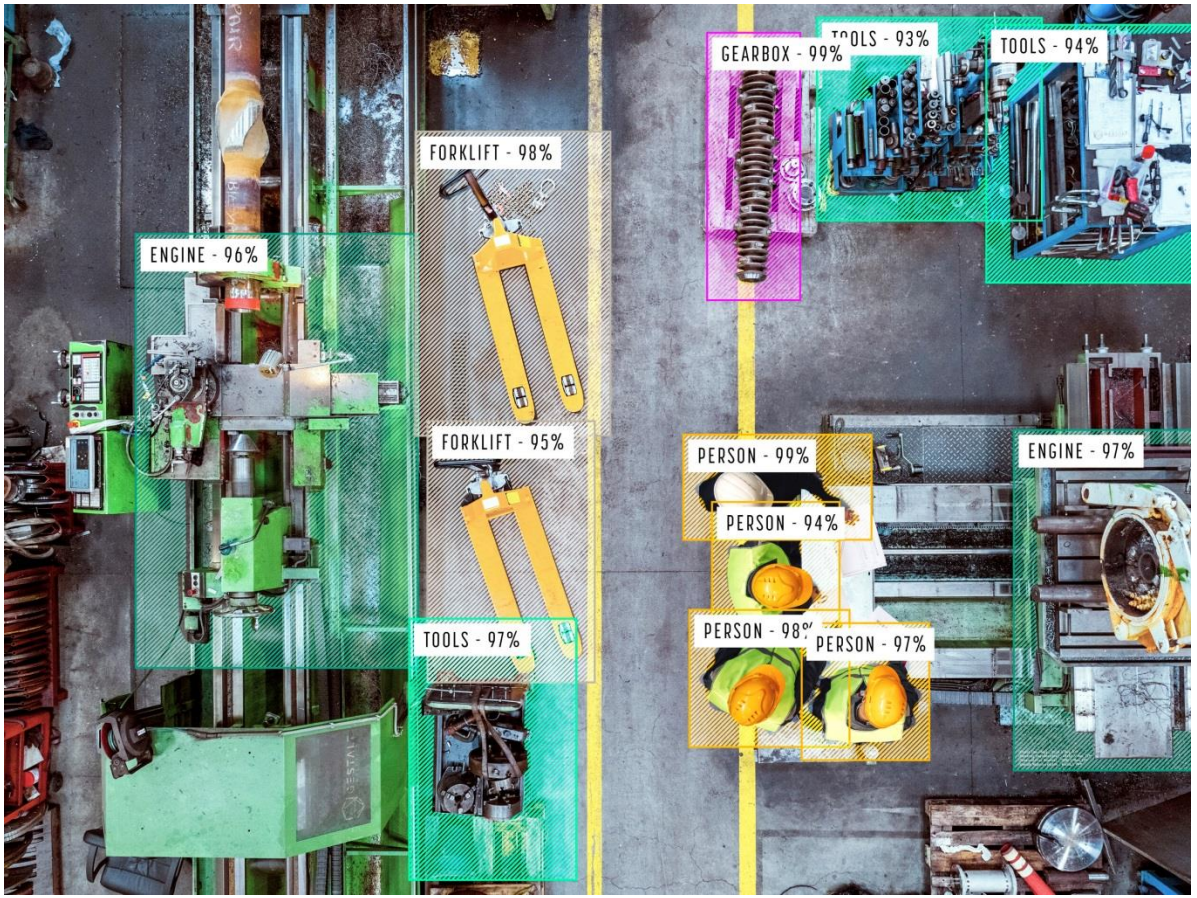


❖ 응용 분야

- 방송 통신 분야
  - 디지털 방송 서비스로 인한 영상처리 기술 발달
  - 스포츠 방송 분야에 영상 처리 기술 적용
  - 가상광고 분야

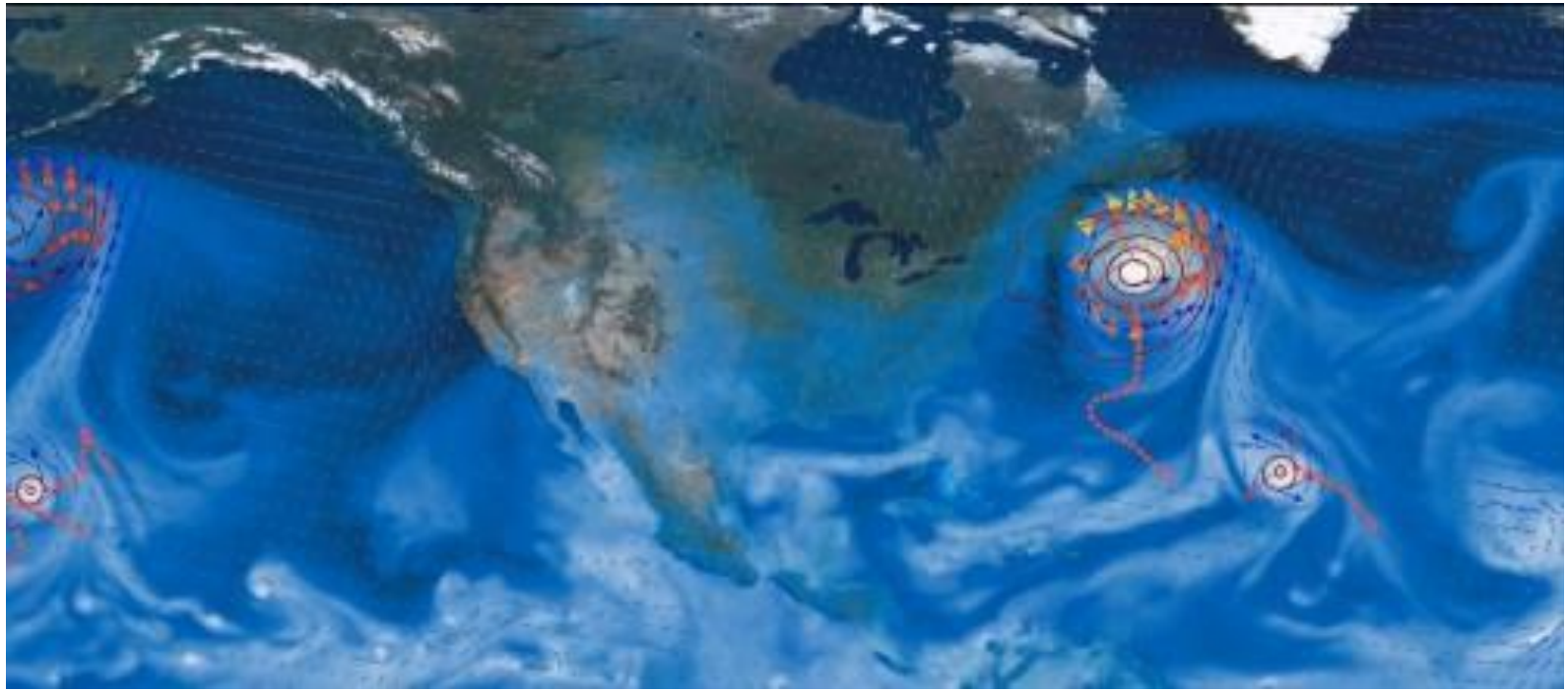
❖ 응용 분야

- 공장 자동화 분야
  - 산업용 카메라로 제품 품질 모니터링 및 불량 제거



❖ 응용 분야

- 기상 및 지질 탐사 분야
  - 방대한 기상 정보를 이용의 시각화
  - 다양한 주파수의 사진들을 영상 처리 기술로 표현



❖ 응용 분야

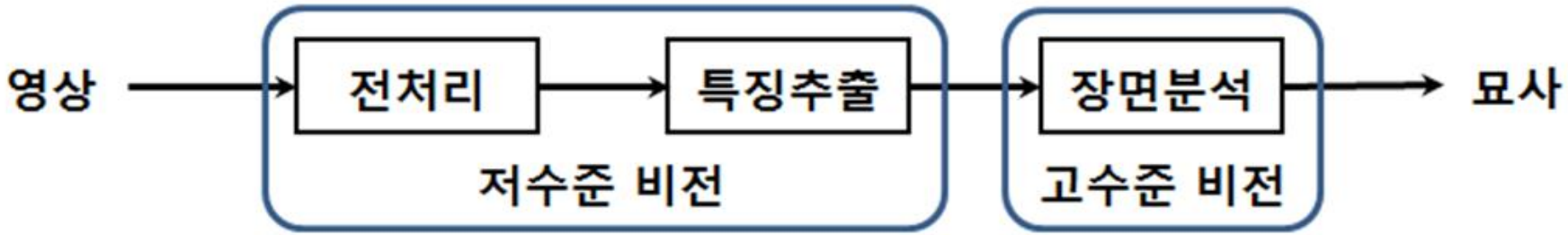
- 애니메이션 및 게임 분야
  - 촬영된 영상과 그래픽 기술이 조합
  - 현실감 향상

❖ 응용 분야

- 출판 및 사진 분야
  - 영상 생성, 품질 향상, 색상을 조작 등의 작업을 위해 영상 처리 기술 사용
  - 기존 영상에 영상 처리 기술을 융합하여 새로운 합성 영상

❖ 컴퓨터 비전 처리 단계

- 전처리 단계
  - 주로 영상처리 기술 사용
  - 다양한 특징 추출 : 에지(edge), 선분, 영역, SIFT(Scale-Invariant Feature Transform) 등
- 고수준 처리
  - 특징정보를 사용하여 영상을 해석, 분류, 상황묘사 등 정보 생성

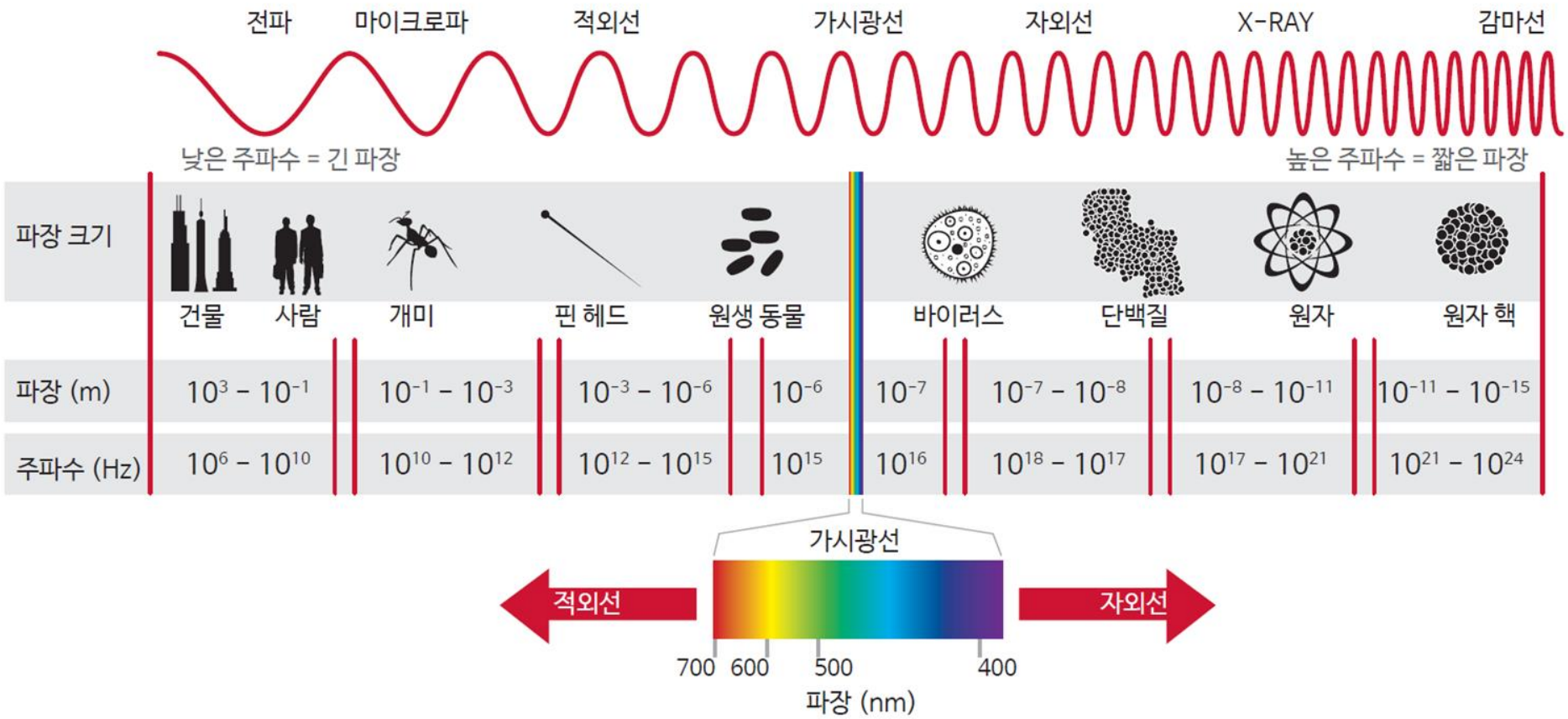


❖ 화소(pixel)

- 디지털 영상을 표현하는 2차원 배열에서 각 원소
- 해당 위치에서 빛의 세기에 대응하는 값
  - 0은 검은색을 나타내고, 화소값이 커질수록 밝은 색
- 컬러 영상
  - R(red), G(green), B(blue) 세 가지 색상에 대한 정보 화소 정보 표현
  - 2차원 행렬 3개로 표현
- 화소를 처리하는 것이 영상 처리의 시작

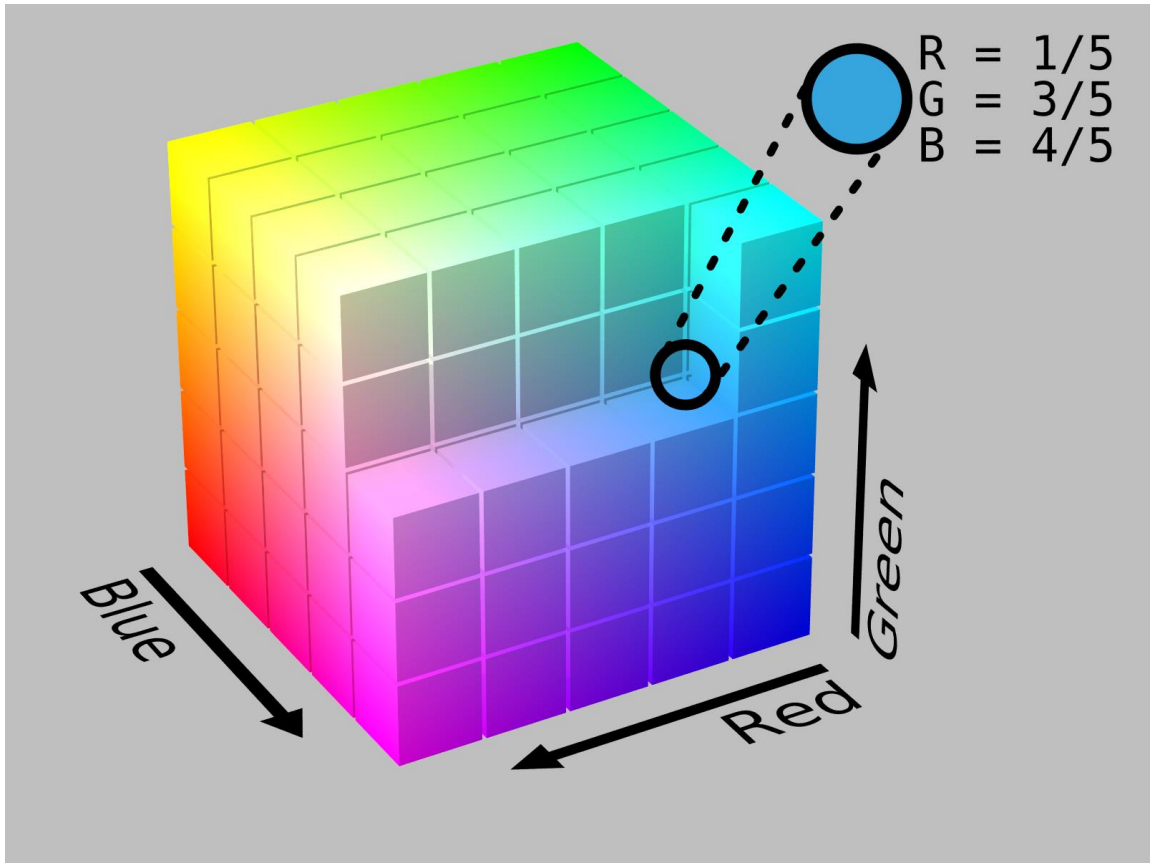
❖ 이미지와 색공간

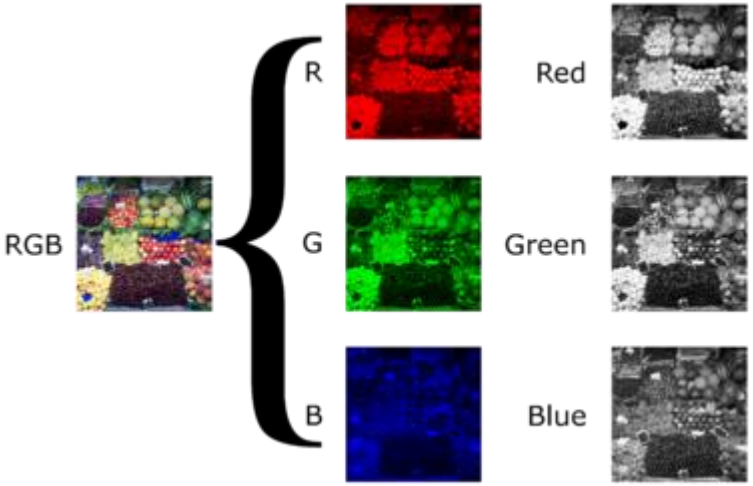
- 색: 빛에서 주파수(파장)의 차이에 따라 다르게 느껴지는 색상
- 가시광선: 전자기파 중에서 인간이 인지할 수 있는 약 380nm~780nm 사이의 파장



❖ 이미지와 색공간

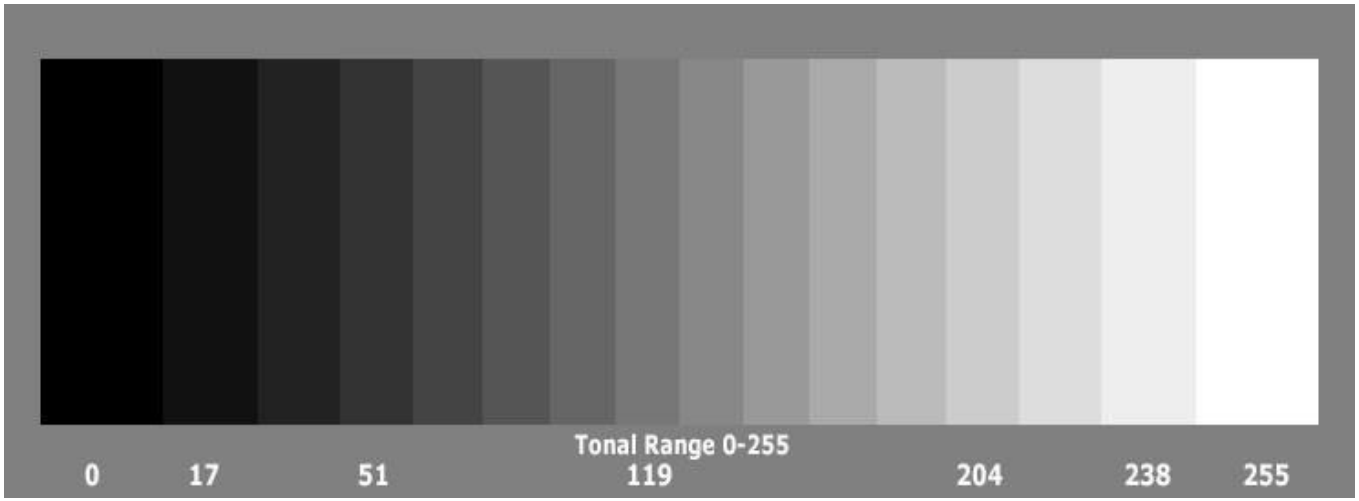
- 0 ~ 255 사이의 값으로 밝기를 표현
- color: 3차원 (true color 라고도 불림)

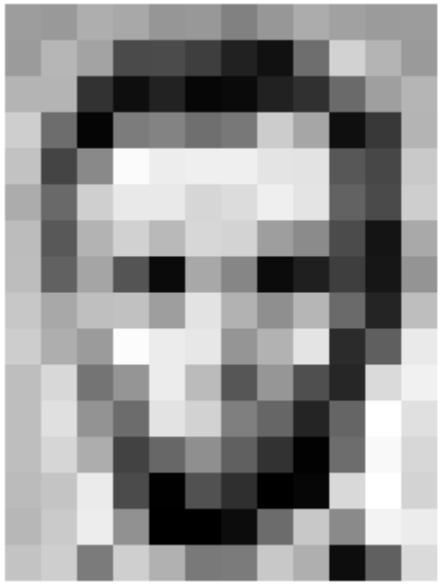




❖ 이미지와 색공간

- gray scale: 2차원





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

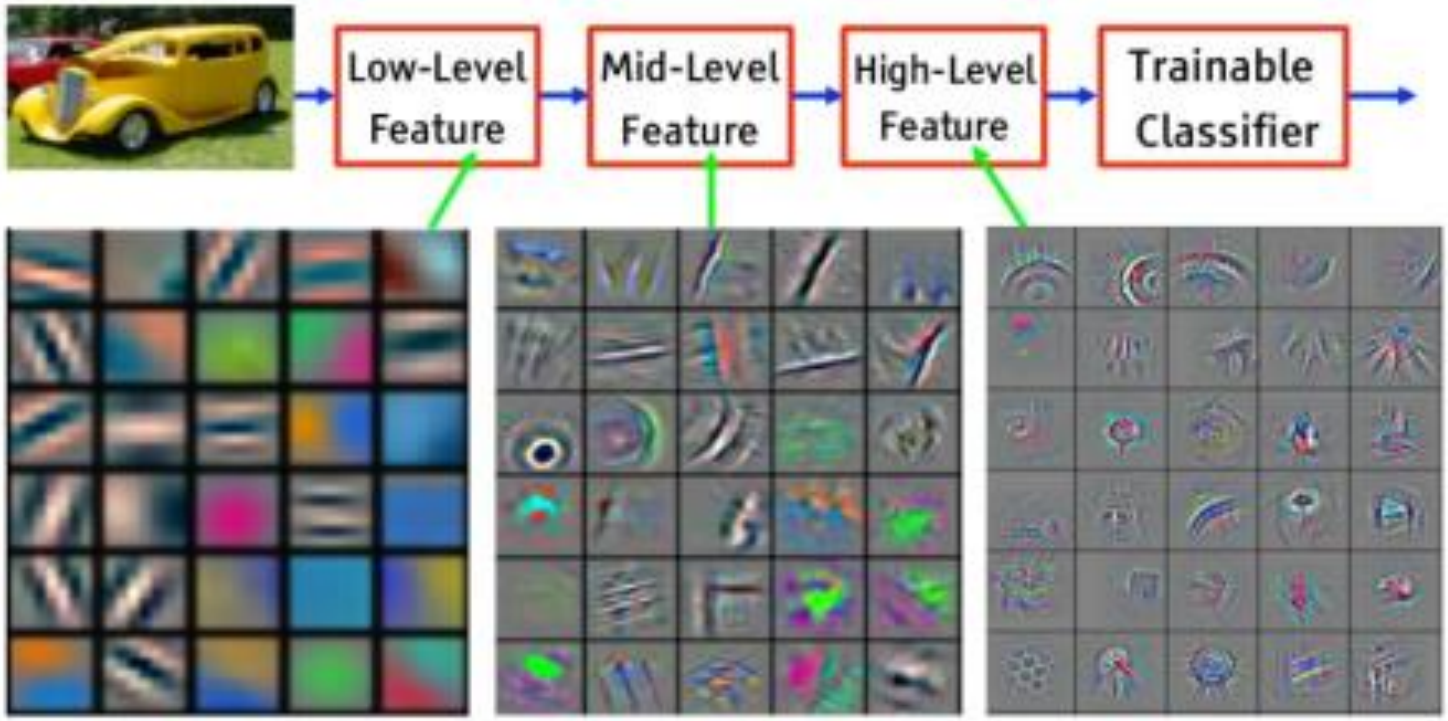
❖ 이미지 파일 형식

- BMP
  - 픽셀 데이터를 압축하지 않은 상태로 저장
  - 파일 구조 간단하지만 용량이 매우 큼
- JPG(JPEG)
  - 손실 압축(lossy compression) 사용
  - 원본 영상으로부터 픽셀값이 미세하게 달라짐
  - 파일 용량 크기가 크게 감소하는 점에서 장점
  - 디지털 카메라
- GIF
  - 무손실 압축(losses compression)
  - 움직이는 그림인 Animation GIF 지원
  - 256가지 이하의 색상을 가진 영상만을 저장하고, 화질이 매우 떨어짐
- PNG(Portable Network Graphics)
  - 무손실 압축 사용
  - 용량은 큰 편이지만 픽셀값이 변경되지 않음
  - α 채널을 지원하여 일부분을 투명하게 설정 가능

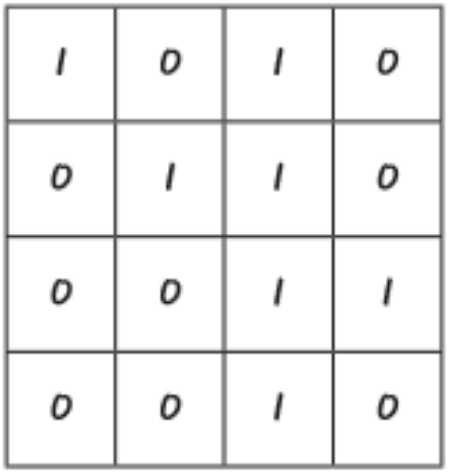
2. 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 개요

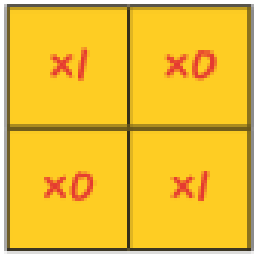
- 이미지 처리에 탁월한 성능을 보이는 신경망
- 이미지 전체를 작은 단위로 쪼개어 각 부분을 분석하는 것이 핵심
- 이미지를 인식하기 위해 패턴을 찾는 데 유용함
- 컨볼루션, 활성화, 서브샘플링 과정을 반복함으로써 저차원적인 특성부터 시작해서 고차원적인 특성을 도출해나간 후 최종 특성을 가지고 분류작업을 실시



❖ Mask(Filter, Window, Kernel)

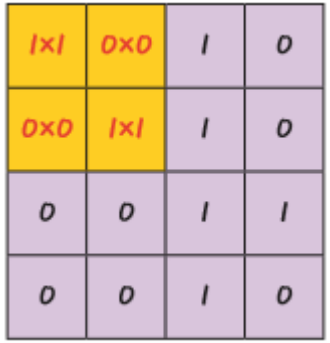


주어진 이미지

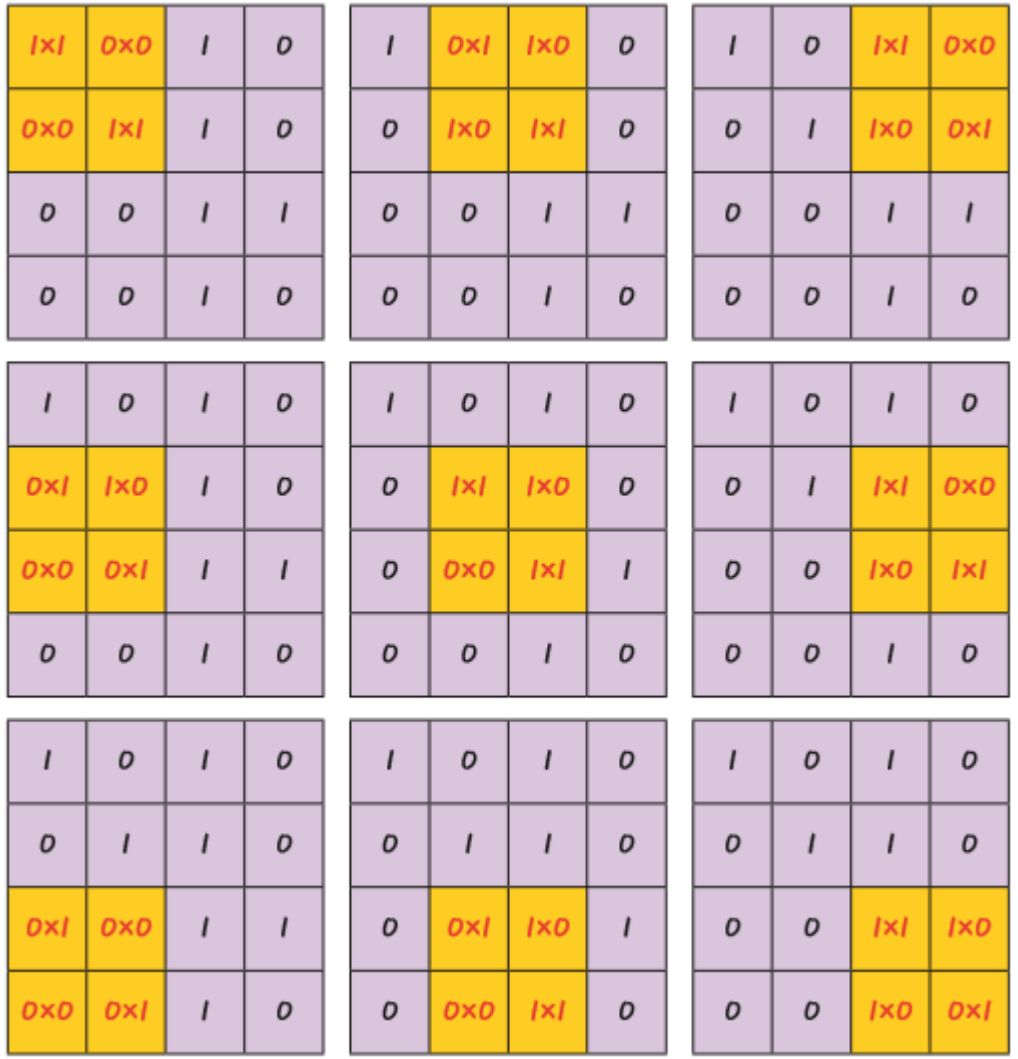


필터

❖ Convolution 과정



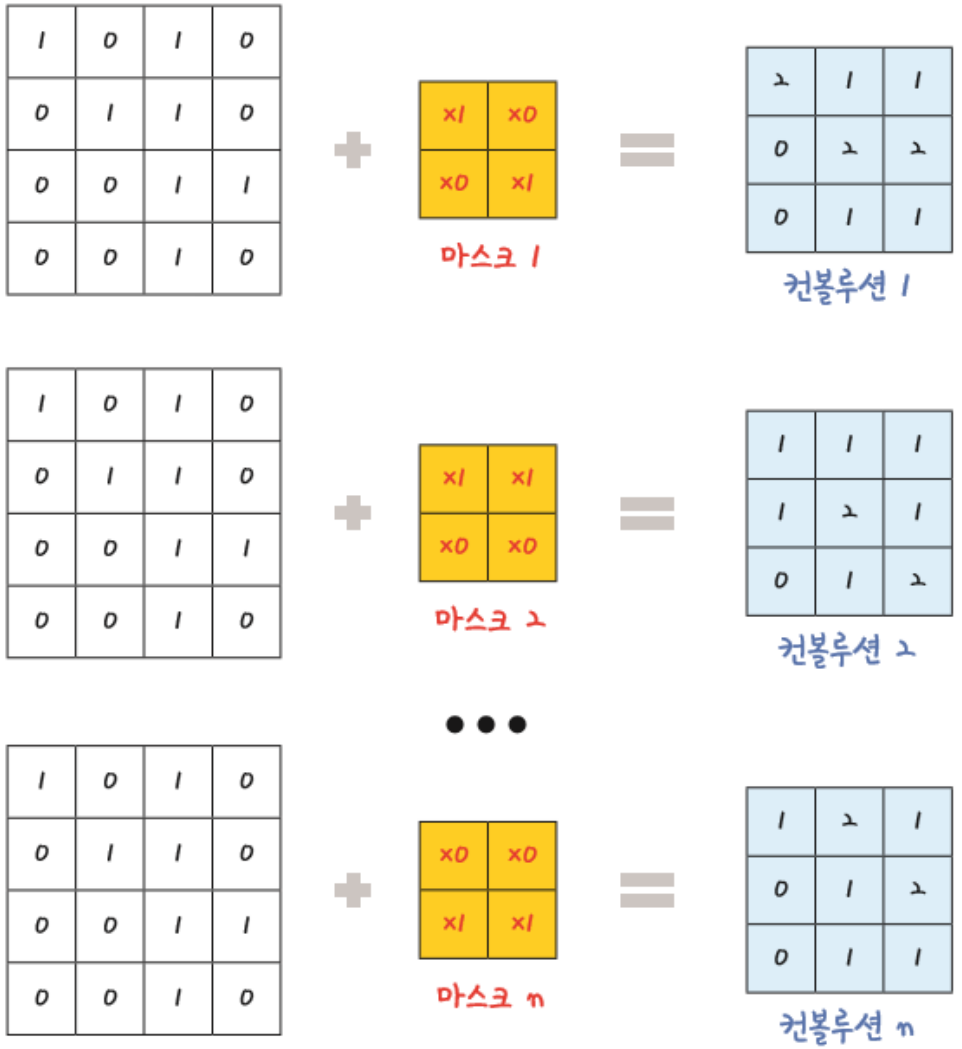
$$(1\times1)+(0\times0)+(0\times0)+(1\times1)=2$$



❖ Convolution 과정

- 컨볼루션을 만들면 입력 데이터로부터 더욱 정교한 특징을 추출할 수 있음

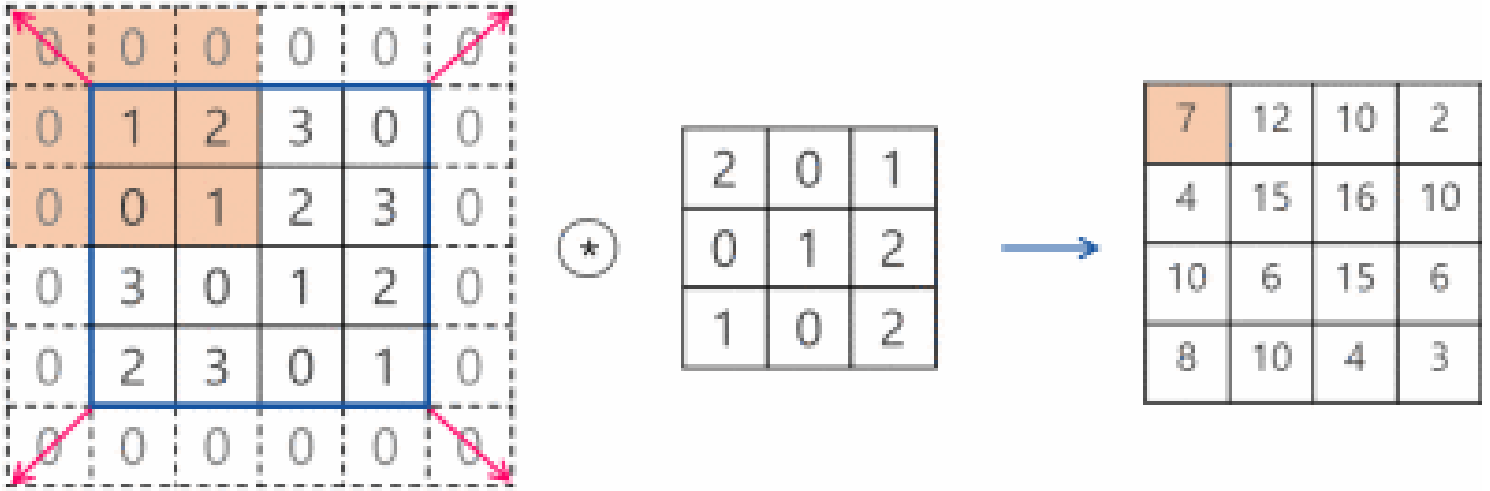
- 이러한 마스크를 여러 개 만들 경우 여러 개의 컨볼루션이 만들어짐 (Feature Map)



2. 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Padding, Stride

- 패딩: 합성곱 연산을 수행하기 전, 입력데이터 주변을 특정값으로 채워 늘리는 것
- 스트라이드: 입력데이터에 필터를 적용할 때 이동할 간격을 조절하는 것

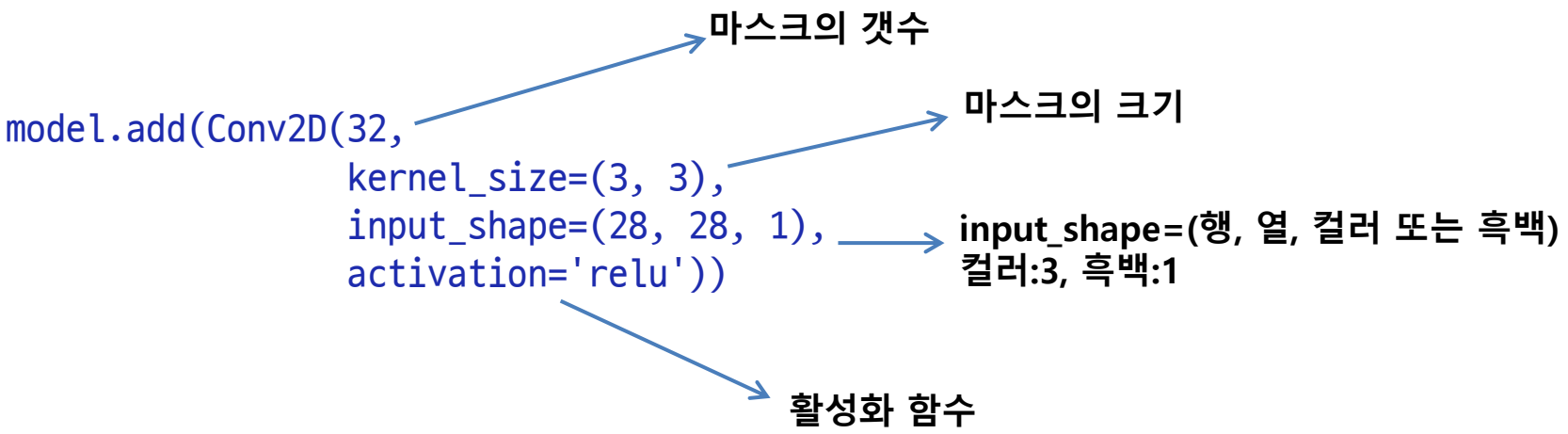


- 출력 이미지의 크기

$$int((N + 2p - f) / s) + 1$$

❖ Keras Convolution 층

- Conv2D

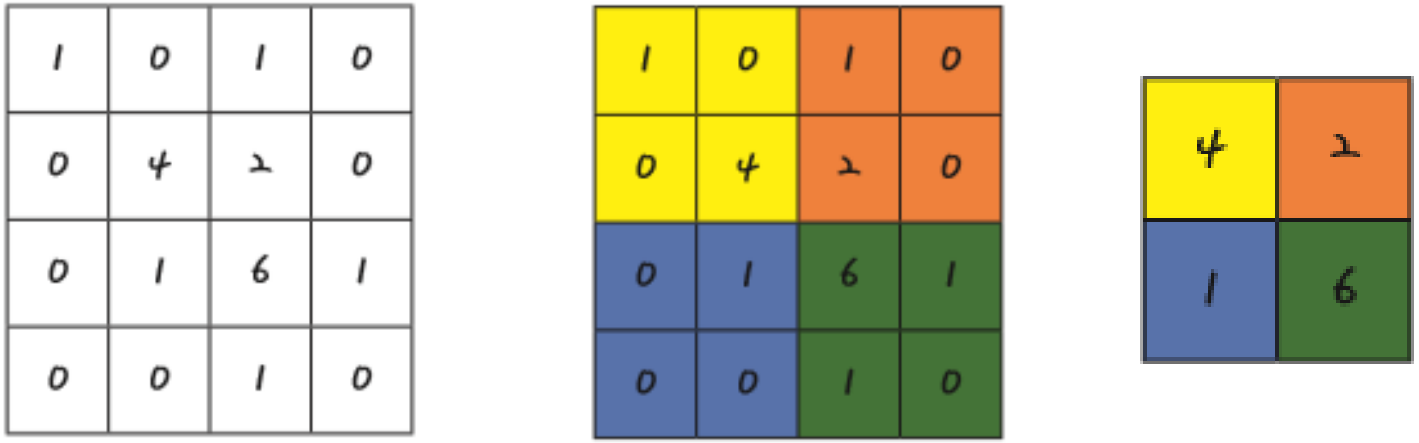


- 파라메터 개수

  입력채널 수 x 마스크 폭 x 마스크 높이 x 출력채널 수(노드 개수) + 출력채널 수

❖ 풀링(Pooling)

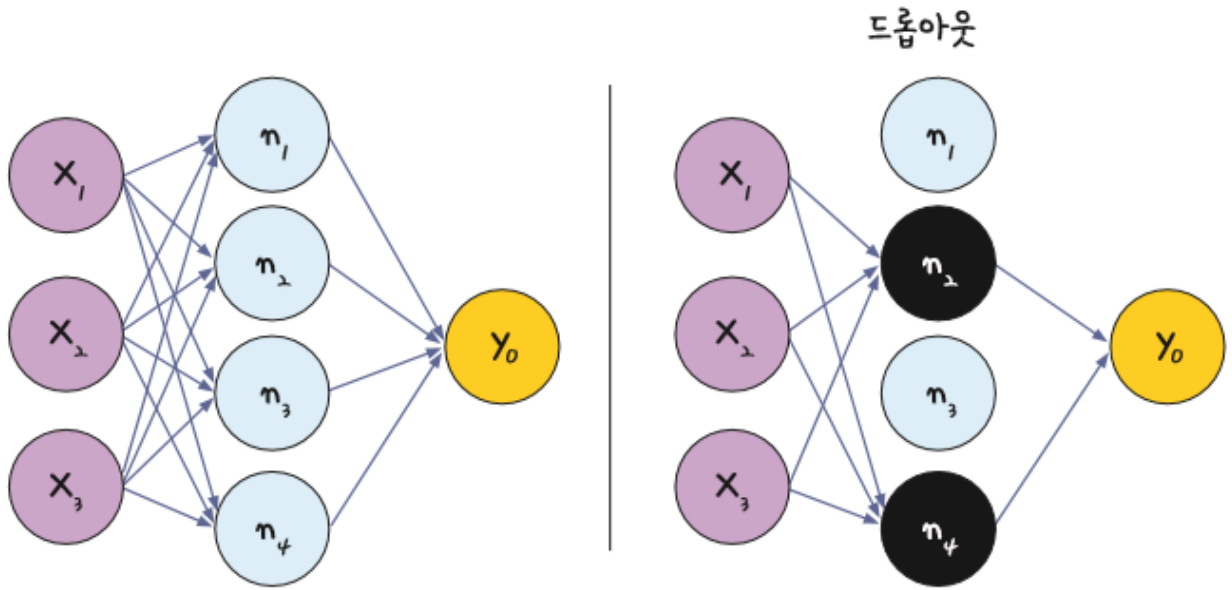
- Convolution 결과를 축소하는 것
- 풀링 기법 중 가장 많이 사용되는 방법이 맥스 풀링(max pooling)
- 맥스 풀링은 정해진 구역 안에서 가장 큰 값만 다음 층으로 넘기고 나머지는 버림




```
model.add(MaxPooling2D(pool_size=2))
```

❖ 드롭아웃(Drop out)

- 노드가 많아지거나 층이 많아진다고 해서 학습이 무조건 좋아지는 것이 아니다
  → 과적합 발생
- 과접합을 피하는 간단하지만 효과가 큰 기법이 바로 드롭아웃(drop out) 기법
- 드롭아웃은 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 것




```
model.add(Dropout(0.25))   # 25%의 노드를 끄기
```

❖ 플래튼(Flatten)

- 콘볼루션, 맥스풀링, 드롭아웃 층을 거친 후 기본 층에 연결
  - 콘볼루션, 맥스풀링: 2차원
  - 기본 층: 1차원
- 2차원 → 1차원 변환

```
model.add(Flatten())
```

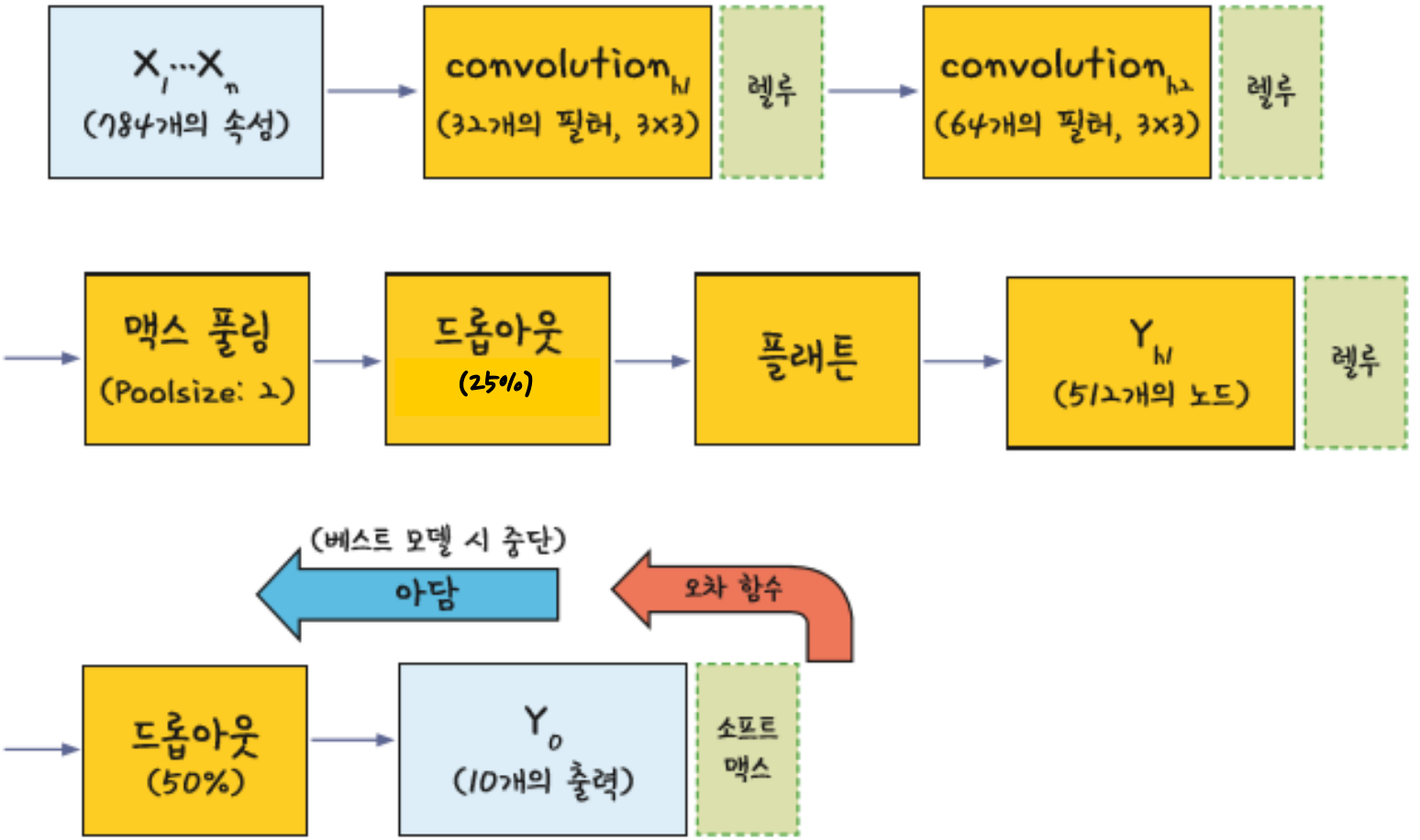
2. 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ MNIST 손글씨 CNN 모델 예

- 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손글씨를 이용해 만든 데이터
- 70,000개의 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터셋

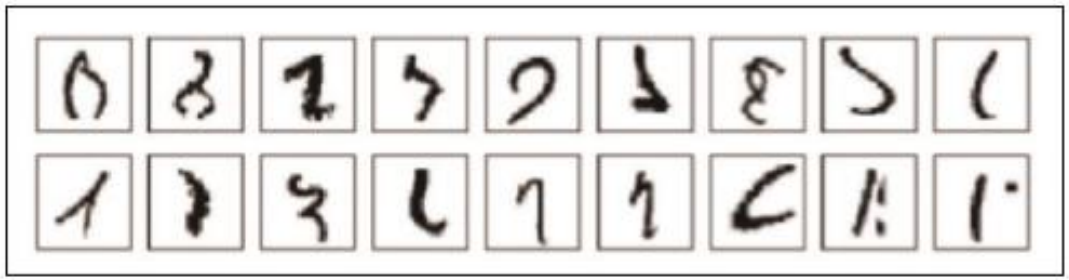
2. 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ MNIST 손글씨 CNN 모델 예



❖ MNIST 손글씨 CNN 모델 결과 리뷰

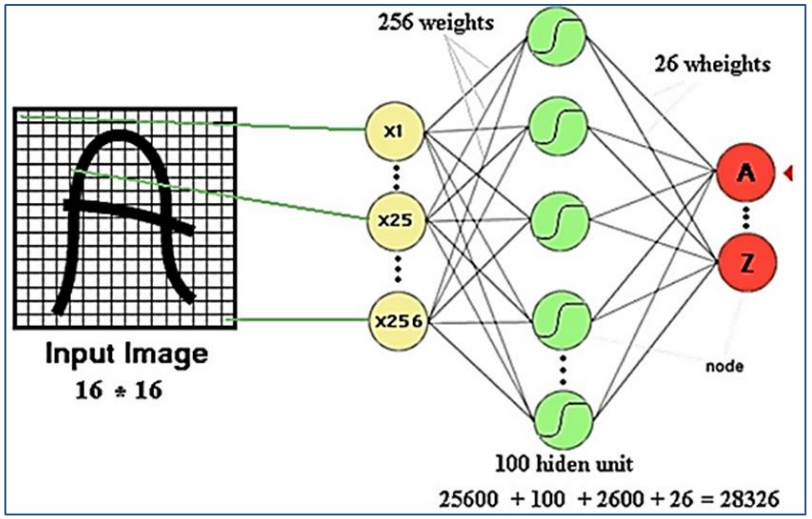
- 99% 대의 정확도
- 심층 신경망 코드에서는 정확도가 97% 대
- 100% 다 맞히지 못한 이유는 데이터 안에 다음과 같이 확인할 수 없는 글씨가 들어있었기 때문

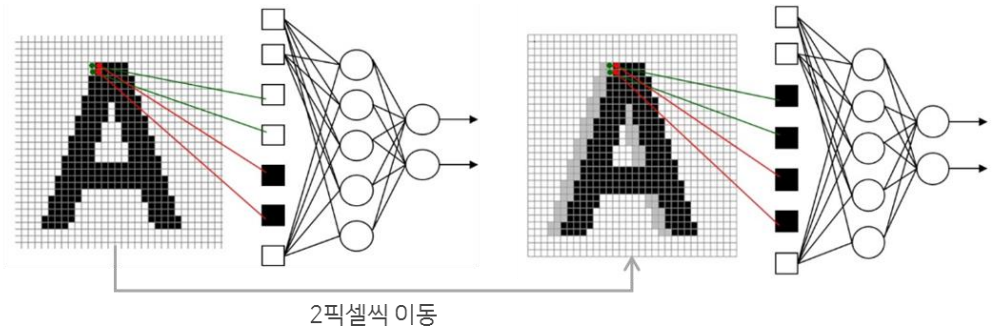


2. 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 심층 신경망(DNN)으로 구현했을 때 문제점

- 글자의 형상은 고려하지 않고, 글자의 크기, 회전, 변형에 취약함



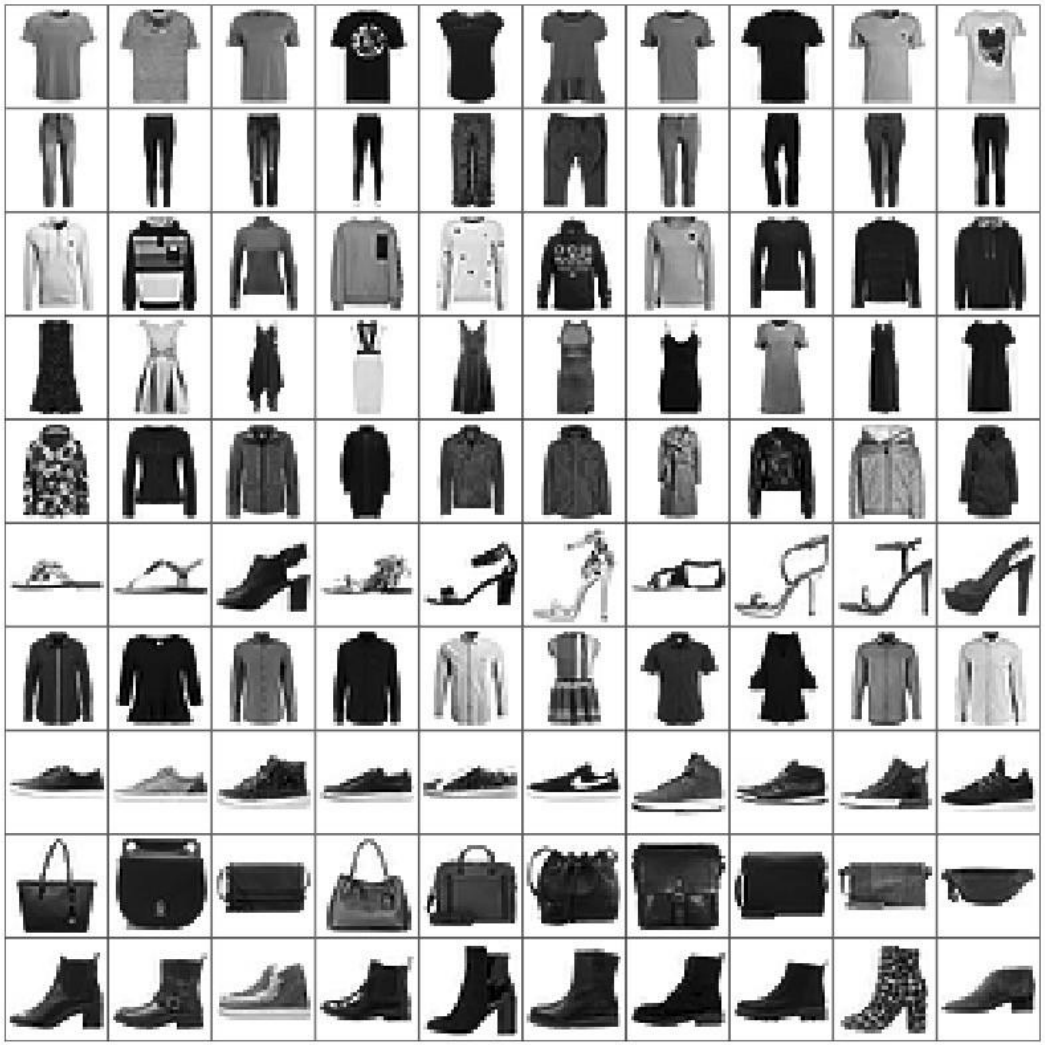


글자의 크기, 회전,
변형에 영향을 받음

→ *글자의 형상은 고려하지 않고, raw data를 직접 처리하기 때문에 많은 양의 학습 데이터가 필요하고, 따라서 학습 시간이 길어짐*

❖ Fashion MNIST

- 손글씨 데이터셋 대용으로 사용 가능



■ 개, 고양이 구분

❖ Kaggle site

- 2010년 설립된 빅데이터 솔루션 대회 플랫폼 회사
- 2017년 Google이 인수 (ZDNet 기사)
- 기업 및 단체에서 Prize를 걸고 데이터와 해결 과제를 등록하면, 데이터 사이언티스트들이 이를 해결하기 위해 모델을 개발하고 경쟁하게 되는 시스템

❖ Kaggle 구성 요소

- Overview: 문제에 대한 간략한 소개와 문제 정의
- Dataset: 예측 모델을 만들기 위해 필요한 데이터셋 및 field에 대한 설명
- Kernels: 다른 사람들이 어떤 모델을 써서 구현을 했는지 힌트를 얻을 수 있고, 또한 내가 구현한 모델이 과연 올바른지에 관해서 코멘트를 주고받을 수 있음
- Discussion: 게시판 역할
- Leaderboard: 모델 예측 정확도 랭킹

■ 개, 고양이 구분

❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

```
datasets
└── dogs-vs-cats
    ├── train
    │   ├── cat.0.jpg
    │   │    ...
    │   ├── cat.12499.jpg
    │   ├── dog.0.jpg
    │   │    ...
    │   └── dog.12499.jpg
    └── test1
        ├── 1.jpg
        │    ...
        └── 12500.jpg
```

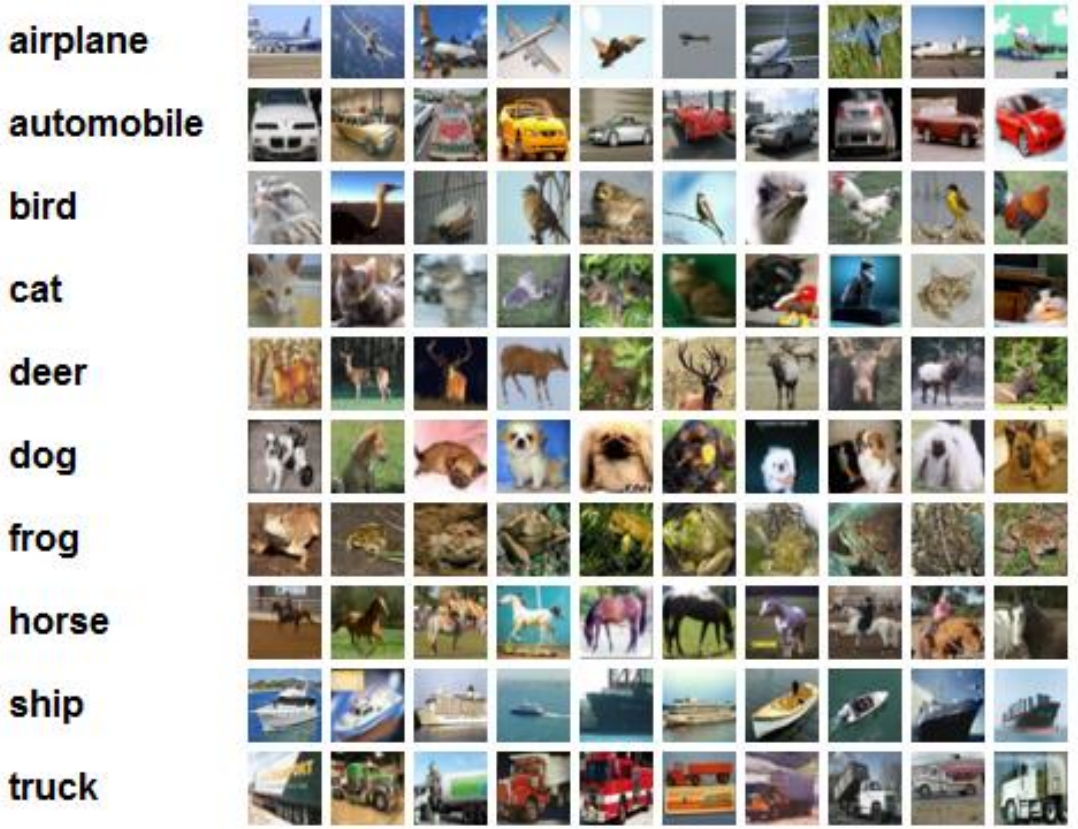
■ 개, 고양이 구분

❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

❖ Cifar 10

- Canadian Institute for Advanced Research
- 32 x 32 크기의 컬러 이미지
- 훈련 셋: 10가지 종류의 50,000개
- 테스트 셋: 10가지 종류의 10,000개



4. 데이터 부풀리기

❖ 데이터 부풀리기(Data Augmentation)

- 원본 이미지에 인위적인 변화를 주어
- 변화된 이미지는 충분히 학습에 활용될 수 있는 데이터가 됨
- 적당한 힘으로 학습 면적을 아주 조금 골고루 넓히자는 의미
- 대부분의 경우 인식의 정확도가 올라감

❖ ImageDataGenerator 클래스

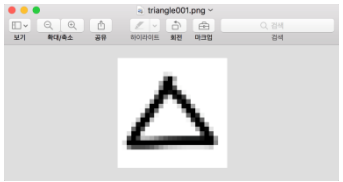
- Keras에서 제공
- 파라메터는 객체 생성시 전달
- *flow_from_directory* 메소드를 활용하면 폴더 형태로된 데이터 구조를 바로 가져와서 사용할 수 있음

❖ ImageDataGenerator 클래스 사용 사례

```
datagen = ImageDataGenerator(
    featurewise_center=False,            # set input mean to 0 over the dataset
    samplewise_center=False,   # set each sample mean to 0
    featurewise_std_normalization=False,           # divide inputs by std of dataset
    samplewise_std_normalization=False,            # divide each input by its std
    zca_whitening=False,       # apply ZCA whitening
    zca_epsilon=1e-06,         # epsilon for ZCA whitening
    rotation_range=0,          # randomly rotate images in the range (deg 0 to 180)
    width_shift_range=0.1,     # randomly shift images horizontally
    height_shift_range=0.1,    # randomly shift images vertically
    shear_range=0.,            # set range for random shear
    zoom_range=0.,             # set range for random zoom
    channel_shift_range=0.,    # set range for random channel shifts
    fill_mode='nearest',       # set mode for filling points outside the input boundaries
    cval=0.,                   # value used for fill_mode = "constant"
    horizontal_flip=True,      # randomly flip images
    vertical_flip=False,       # randomly flip images
    rescale=None,              # set rescaling factor (applied before any other transformation)
    preprocessing_function=None,         # set function that will be applied on each input
    data_format=None,          # image data format, either "channels_first" or "channels_last"
    validation_split=0.0       # fraction of images reserved for validation
)
```

4. 데이터 부풀리기

- 원본 이미지:



- rotation_range = 90, 지정된 각도 범위(90도)내에서 임의로 원본이미지를 회전

- width_shift_range = 0.1, 지정된 수평방향 이동 범위(10%)내에서 임의로 원본이미지를 이동

- height_shift_range = 0.1, 지정된 수직방향 이동 범위(10%)내에서 임의로 원본이미지를 이동

4. 데이터 부풀리기

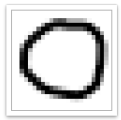
- zoom_range = 0.3, 지정된 확대/축소 범위(0.7 ~ 1.3배)내에서 임의로 원본이미지를 확대/축소

- horizontal_flip = True, 수평방향으로 뒤집기

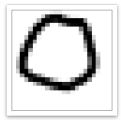
- vertical_flip = True, 수직방향으로 뒤집기

4. 데이터 부풀리기

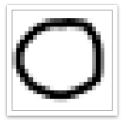
❖ 훈련 셋

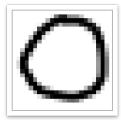

circle001.png

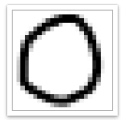

circle002.png

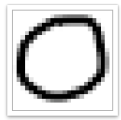

circle003.png

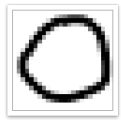

circle004.png

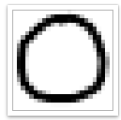

circle005.png

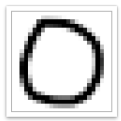

circle006.png

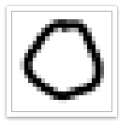

circle007.png


circle008.png

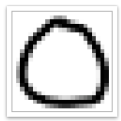

circle009.png

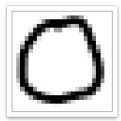

circle010.png

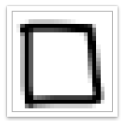

circle011.png

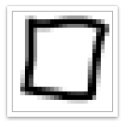

circle012.png

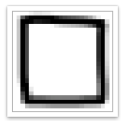

circle013.png

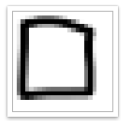

circle014.png

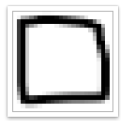

circle015.png

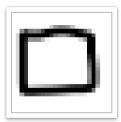

rectangle001.png

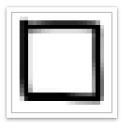

rectangle002.png

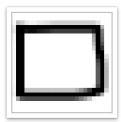

rectangle003.png

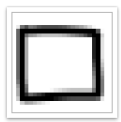

rectangle004.png

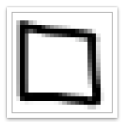

rectangle005.png

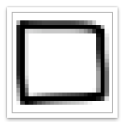

rectangle006.png

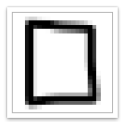

rectangle007.png

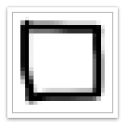

rectangle008.png

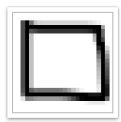

rectangle009.png

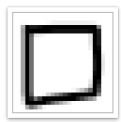

rectangle010.png


rectangle011.png


rectangle012.png


rectangle013.png


rectangle014.png


rectangle015.png


triangle001.png


triangle002.png


triangle003.png


triangle004.png


triangle005.png


triangle006.png


triangle007.png


triangle008.png


triangle009.png


triangle010.png

triangle011.png

triangle012.png

triangle013.png

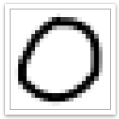
triangle014.png

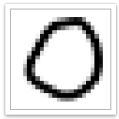
triangle015.png

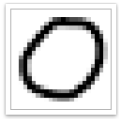
4. 데이터 부풀리기

❖ 테스트 셋

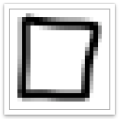

circle016.png

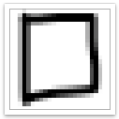

circle017.png

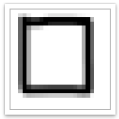

circle018.png

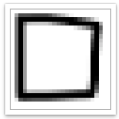

circle019.png

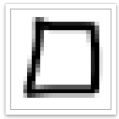

circle020.png


rectangle016.png


rectangle017.png


rectangle018.png


rectangle019.png


rectangle020.png


triangle016.png


triangle017.png


triangle018.png


triangle019.png


triangle020.png

❖ 도전 테스트 셋

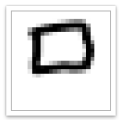

circle021.png


circle022.png


circle023.png


circle024.png


circle025.png


rectangle021.png


rectangle022.png


rectangle023.png


rectangle024.png


rectangle025.png


triangle021.png


triangle022.png

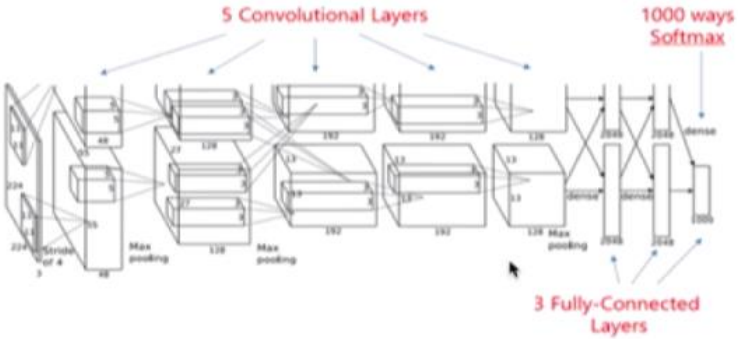

triangle023.png


triangle024.png

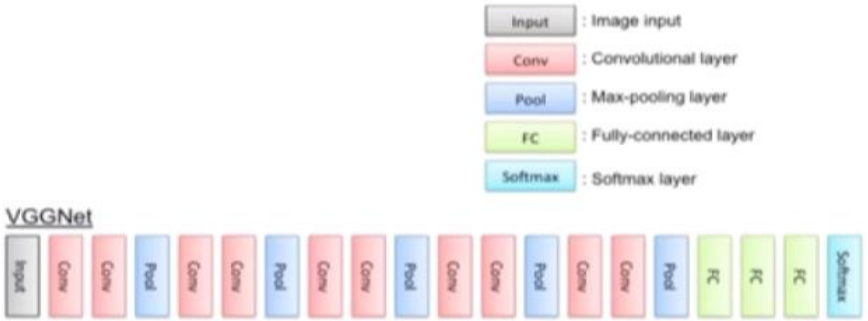

triangle025.png

CNN Architectures

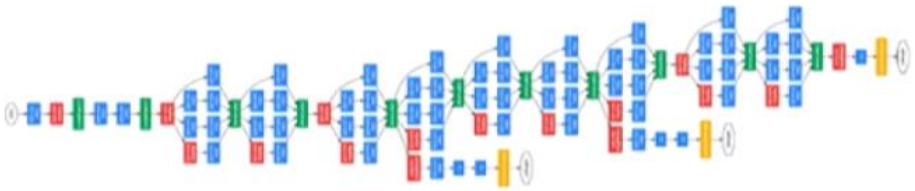
AlexNet



VGG



GoogLeNet



ResNet

5. CNN 주요 모델

❖ AlexNet

- ImageNet에서 주관하는 ILSVRC (Large Scale Visual Recognition Competition) 대회에서, 2012년 제프리 힌튼 교수팀의 AlexNet이 top 5 test error(5개의 예측값 중에 정답이 없는 경우) 기준 15.4%를 기록해 2위(26.2%)를 큰 폭으로 이기고 1위를 차지함.

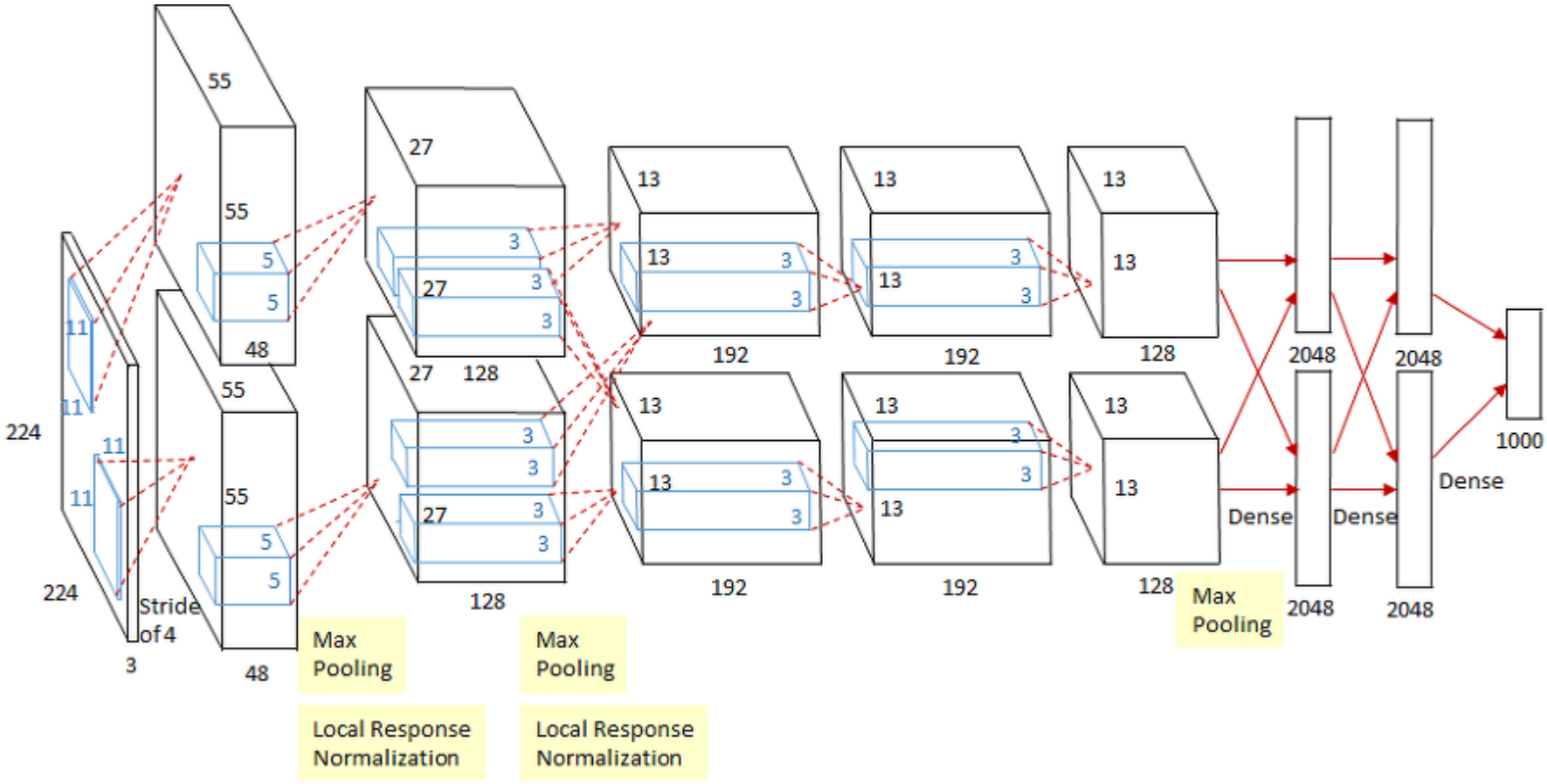
- 이 대회는 1000개의 클래스를 가진 120만장의 이미지를 학습하고 15만장의 이미지로 테스트하여 정답률을 겨루는 대회

- AlexNet의 등장은 딥러닝, 특히 CNN이 본격적으로 주목받게 되는 계기가 되었고 여기서 소개된 ReLU, Dropout 등은 지금도 표준으로 사용되고 있음.

5. CNN 주요 모델

❖ AlexNet

- 구조

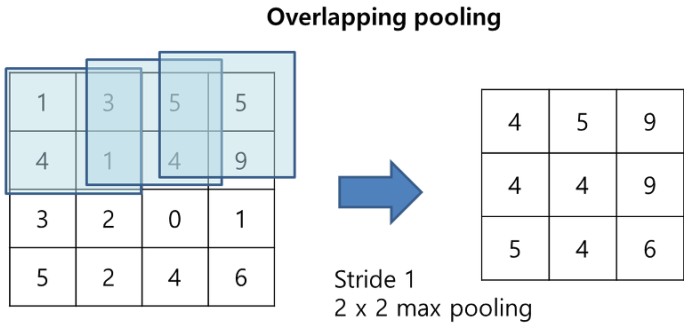


❖ AlexNet

▪ 특징

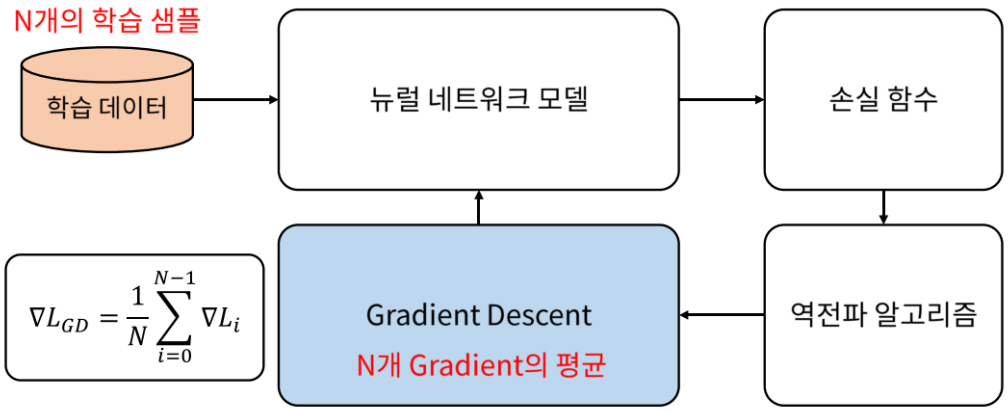
- 5개의 컨볼루션 레이어, 3개의 Fully Connected 레이어로 구성
- 2개의 GPU로 병렬연산 수행

- ReLU 활성화 함수 사용
- Dropout 사용
- Max pooling, Overlapping pooling
- LRN(Local Response Normalization)
- Data Augmentation
- Stochastic Gradient Descent



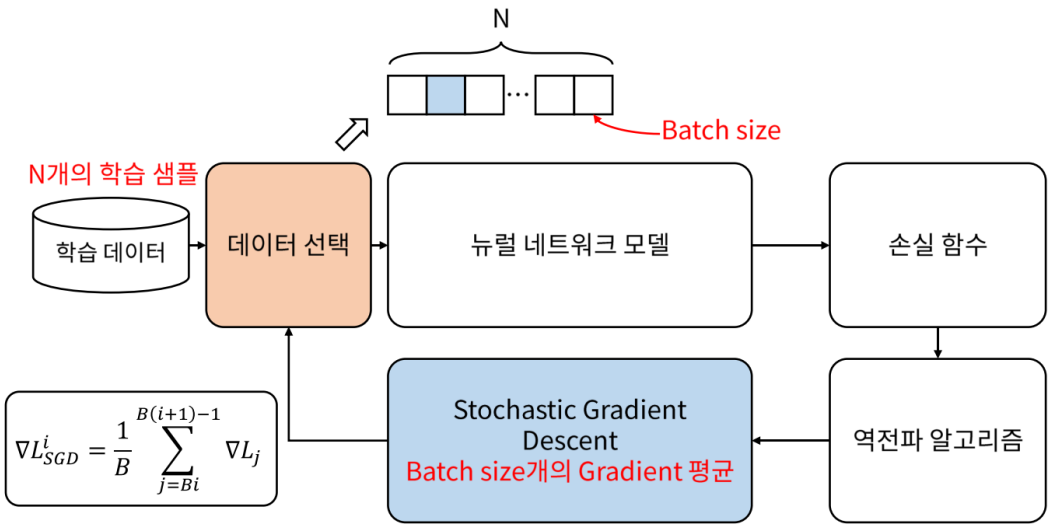
❖ 배치 정규화(Batch Normalization)

▪ 배치(Batch)



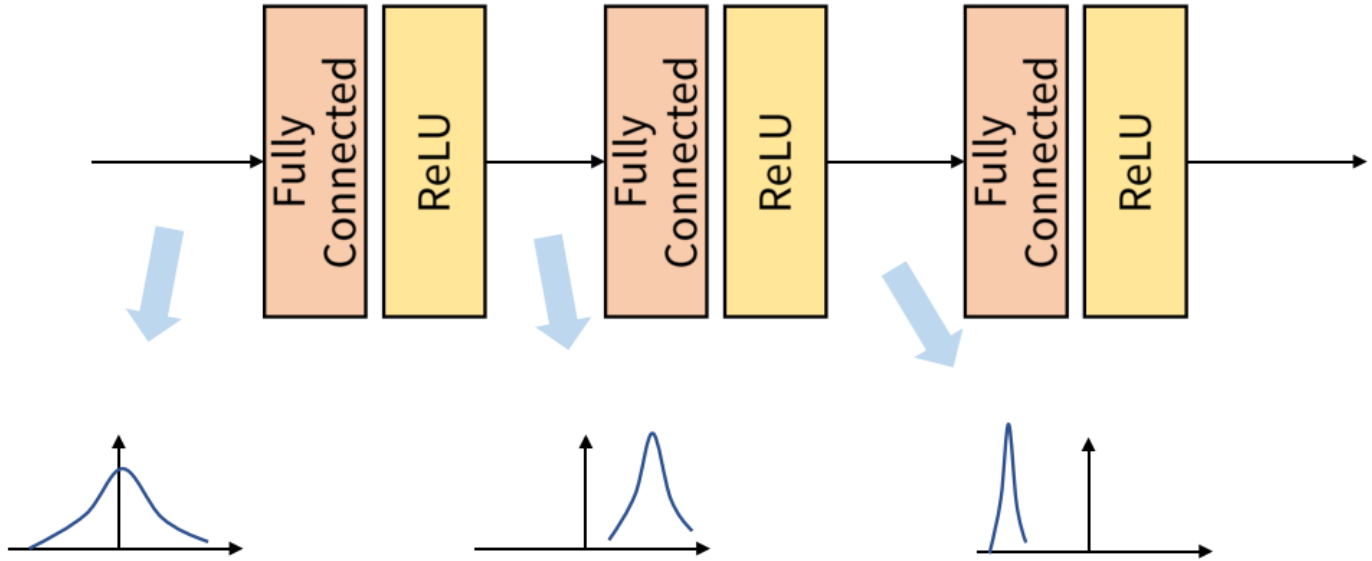
- 학습 데이터 전부를 넣어서 gradient를 다 구하고 그 모든 gradient를 평균해서 한번에 모델 업데이트를 수행
- 이런 방식으로 하면 대용량의 데이터를 한번에 처리하지 못하기 때문에 데이터를 batch 단위로 나눠서 학습을 하는 방법을 사용하는 것이 일반적

- SGD에서는 gradient를 한번 업데이트 하기 위하여 일부의 데이터만을 사용함. 즉, batch size 만큼만 사용



❖ 배치 정규화(Batch Normalization)

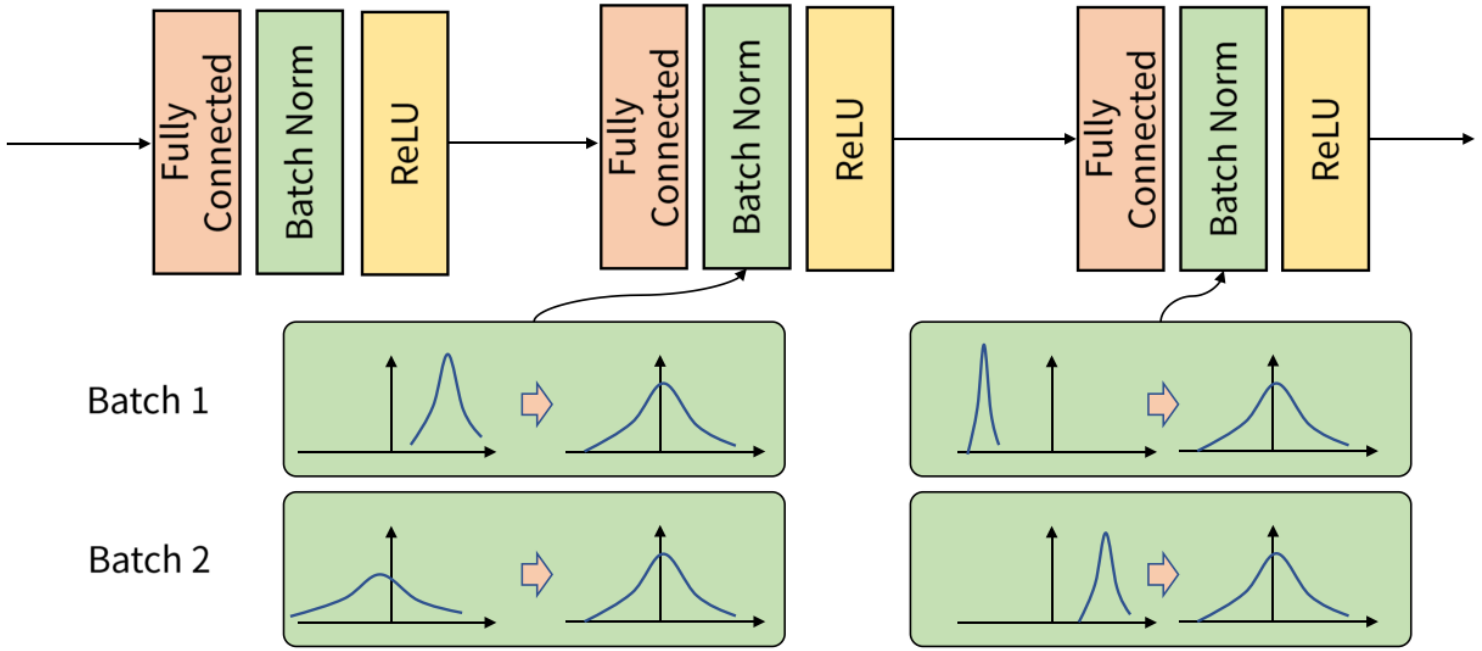
- Internal Covariant Shift



- 위 그림같이 학습 과정에서 계층 별로 입력의 데이터 분포가 달라지는 현상
- 각 계층에서 입력으로 feature를 받게 되고 그 feature는 convolution이나 위와 같이 fully connected 연산을 거친 뒤 activation function을 적용하게 됨.
- 그러면 연산 전/후에 데이터 간 분포가 달라질 수가 있음.
- 이와 유사하게 Batch 단위로 학습을 하게 되면 **Batch 단위간에 데이터 분포의 차이**가 발생할 수 있음.

❖ 배치 정규화(Batch Normalization)

- Batch Normalization



- 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화하는 것

❖ 배치 정규화(Batch Normalization)

- Batch Normalization

$$BN(X) = \gamma\left(\frac{X - \mu_{batch}}{\sigma_{batch}}\right) + \beta$$

$$\mu_{batch} = \frac{1}{B}\sum_{i} x_i$$

$$\sigma_{batch}{}^2 = \frac{1}{B}\sum_{i}(x_i - \mu_{batch})^2$$

- γ는 스케일링 역할을 하고 β는 bias임
- 두 값 모두 back propagation을 통하여 학습을 하게 됨

❖ VGGNet

- 2014년 ILSVRC 대회에서, 2등을 한 모델

- GoogLeNet에 밀려 2위를 했지만, 훨씬 간단한 구조로 이해와 변형이 쉽다는 장점이 있어 많이 응용되는 모델

- 깊이에 따른 변화를 비교하기 위해, 3x3의 작은 필터 크기를 사용했고, 모델 깊이와 구조에 변화를 주어 실험. (6가지 모델)

❖ VGGNet

- 구조

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

❖ VGGNet

▪ 특징

- Small filters, Deeper networks

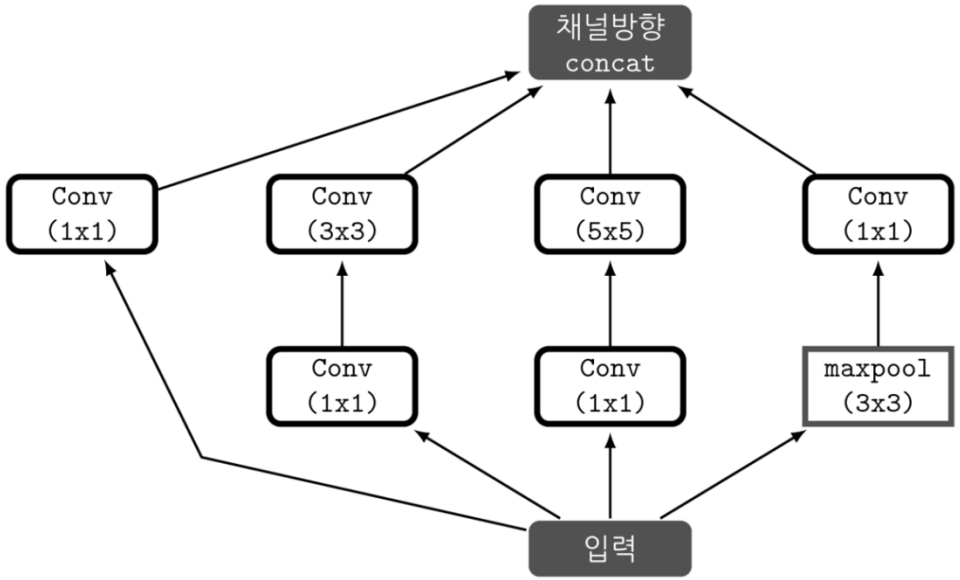
- 8개의 layer를 가지는 AlexNet에서 16~19개의 layer를 가지는 VGGNet으로 발전
- 3x3의 크기를 가지는 filter를 사용
- stride=1, padding=1 인 convolution layer
- 2x2 max pooling with stride=2인 pooling layer

- 3x3 을 깊게 쌓게 되면, 우선 비선형성을 더 많이 반영할 수 있으며, 실제로 필요한 parameter 수도 적게된다는 장점이 있음

❖ GoogLeNet (Inception)

- 2014년 ILSVRC 대회에서, 1등을 한 모델
- 22개의 레이어
- 노드 간의 연결을 줄이면서(Sparse connectivity), 행렬 연산은 Dense 연산을 하도록 처리

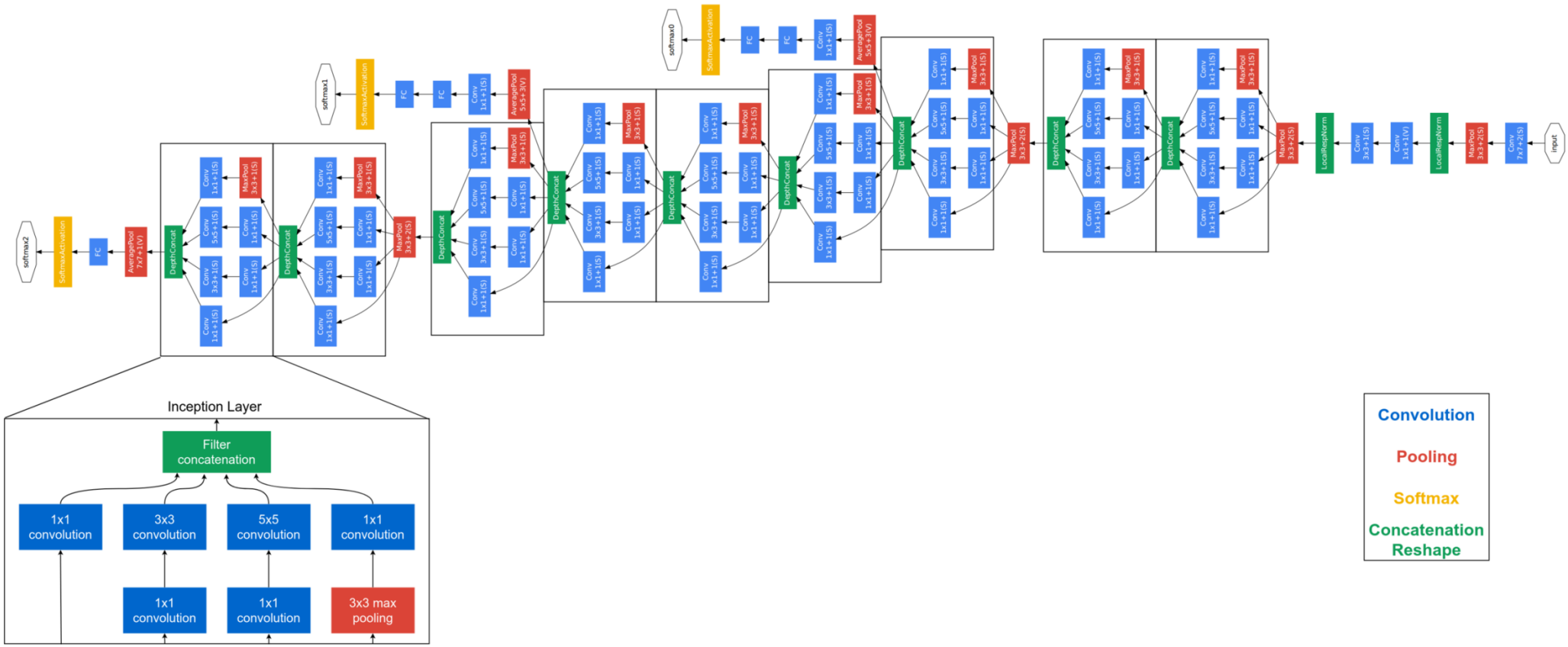
  ➔ Inception module



5. CNN 주요 모델

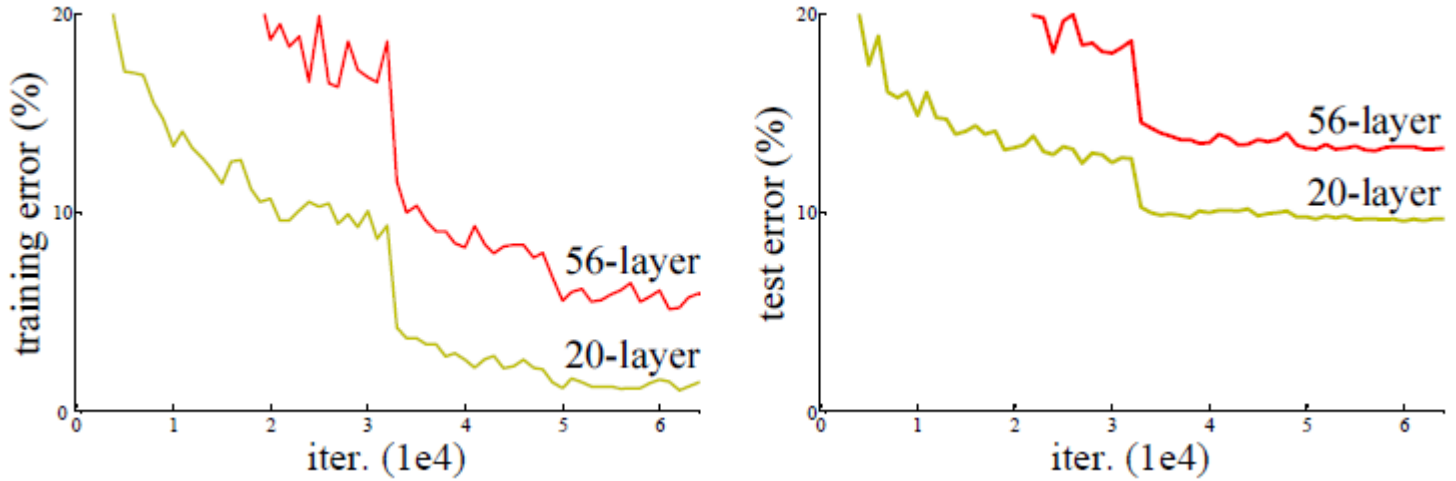
❖ GoogLeNet (Inception)

- 구조



❖ ResNet (Residual Network)

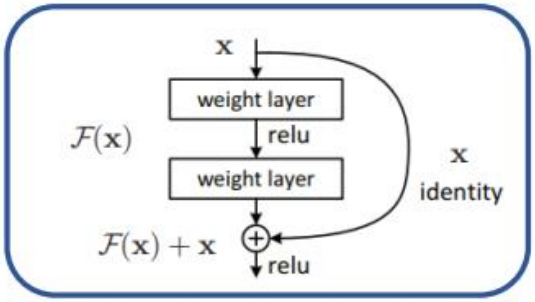
- 2015년 ILSVRC 대회에서, 1등을 한 모델 (Microsoft)
- 152개 층
- 망을 깊게하면 무조건 성능이 좋아질까?

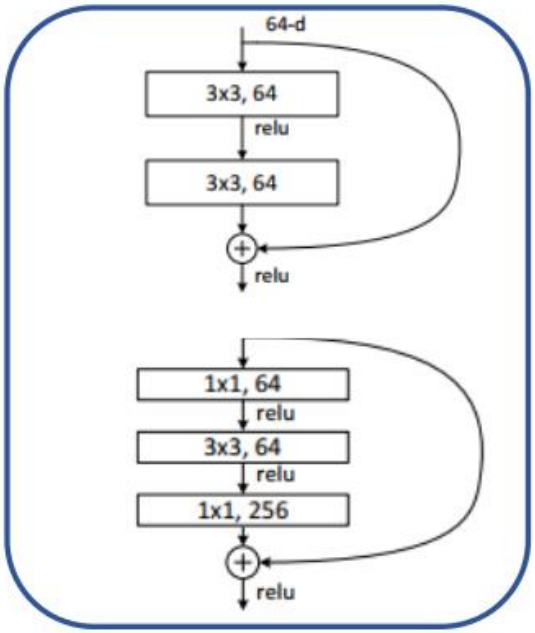


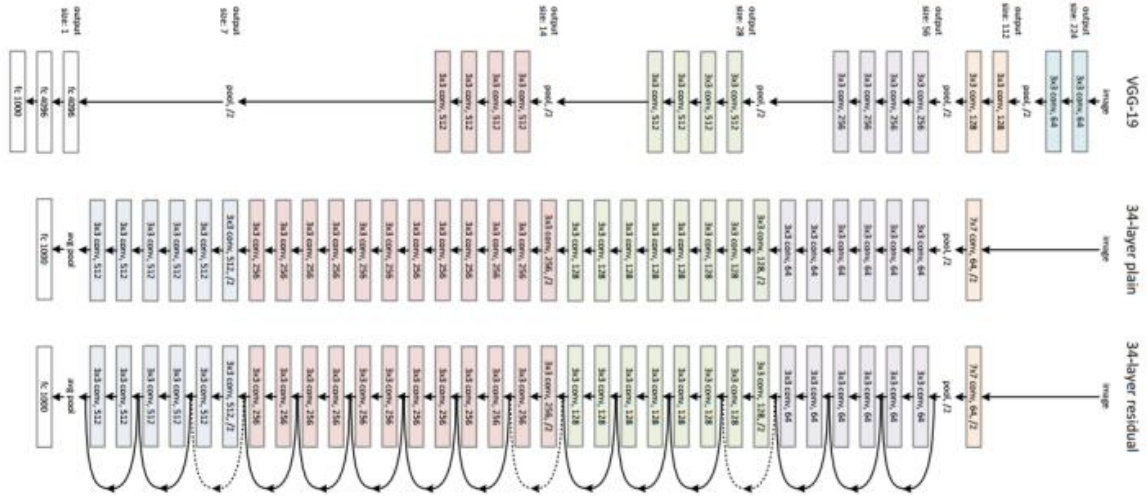
- Degradation 문제 해결

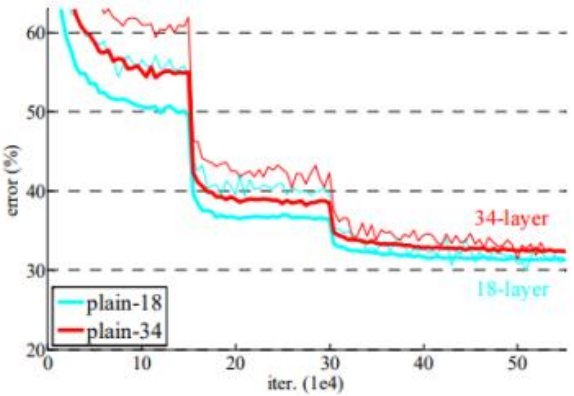
❖ ResNet

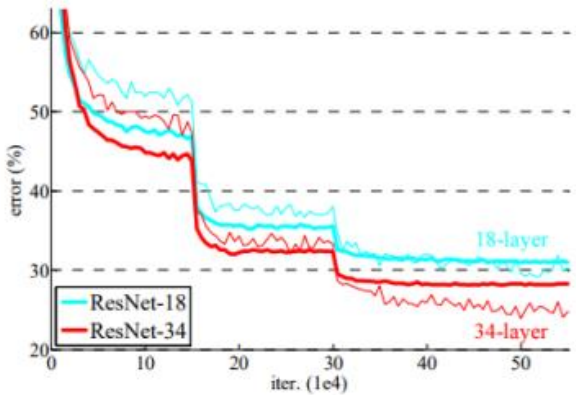
- 구조

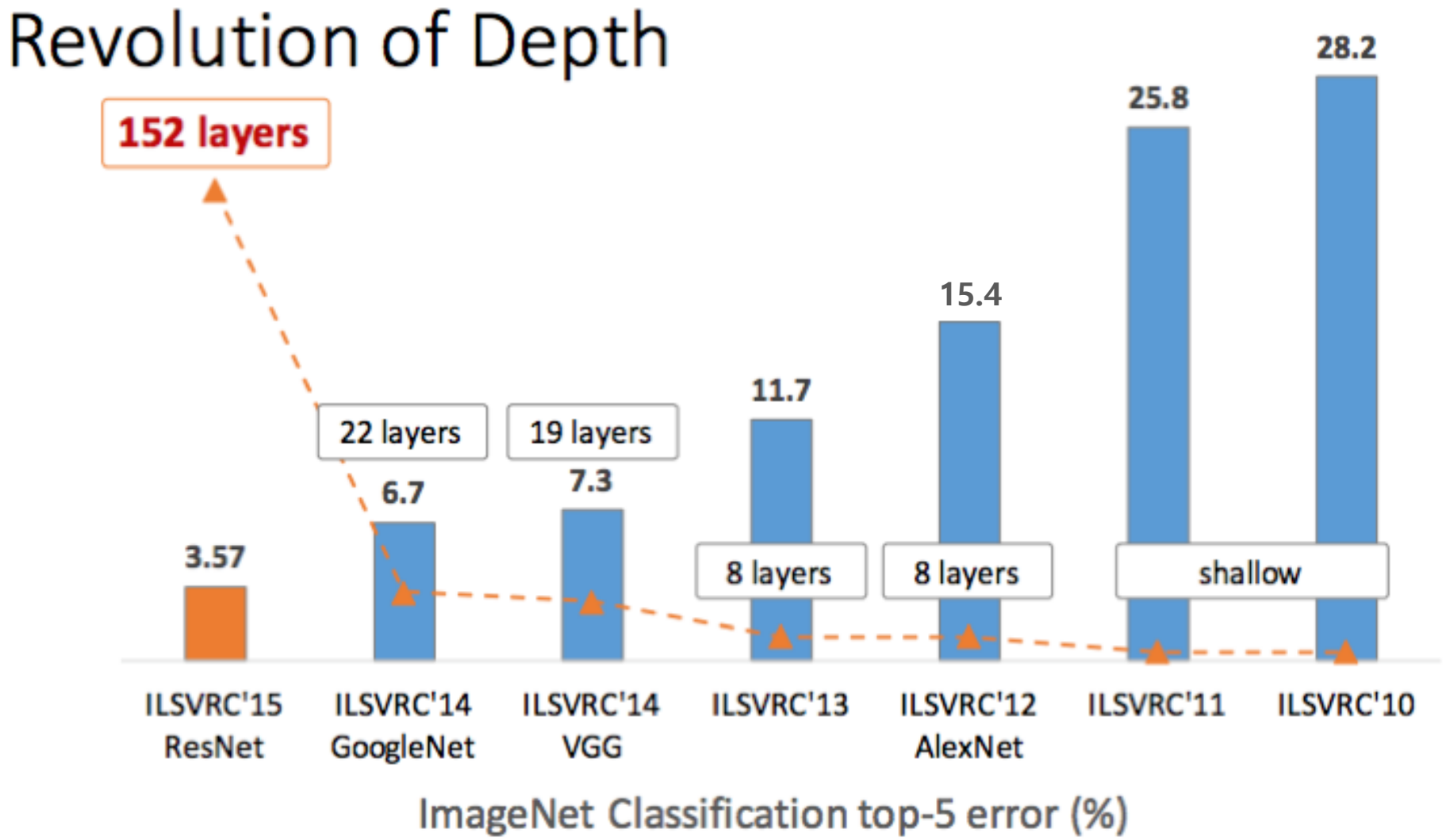

x
weight layer
relu
$\mathcal{F}(x)$
weight layer
x
identity
$\mathcal{F}(x) + x$
relu


64-d
3x3, 64
relu
3x3, 64
relu
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu


VGG-19
34-layer plain
34-layer residual


error (%)
iter. (1e4)
plain-18
plain-34
34-layer
18-layer


error (%)
iter. (1e4)
ResNet-18
ResNet-34
18-layer
34-layer


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
15.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)