

지능화 캡스톤 프로젝트

[프로젝트 #1]

# CNN 영상분류 논문리뷰

2022. 03. 16

김 현 용

충북대학교 산업인공지능학과

# 목차

1. 논문에 대해 ...
2. 영상분류 논문리뷰
3. [참고] 딥러닝 복습

- 목적

- 전형적인 이미지 분류 관련 SCI 논문을 구현함으로써 CNN 구현능력 배양
- 논문리뷰를 통해 학위논문(보고서) 작성법 학습

## A Deep Convolutional Neural Network for Wafer Defect Identification on an Imbalanced Dataset in Semiconductor Manufacturing Processes

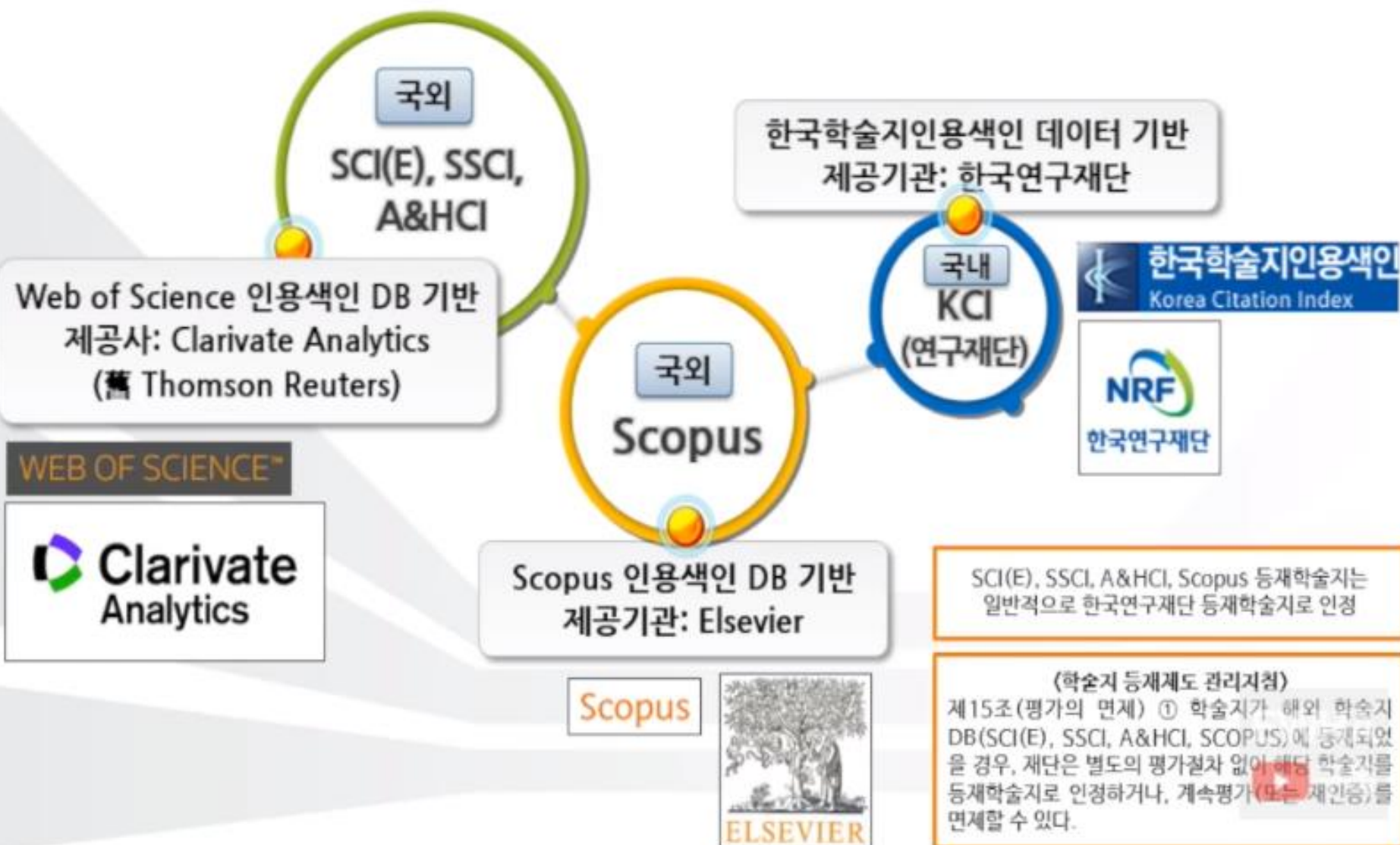
Muhammad Saqlain<sup>1</sup>, Qasim Abbas, and Jong Yun Lee

반도체 제조공정의 불균형 데이터셋에 대한 웨이퍼 불량 식별을 위한 심층  
컨볼루션 신경망

무하매드 사칼린, 카심 아바스, 이종연

# 학술지의 등재정보

## • 국내외 주요 학술지 등재정보



# SCI 학술지의 평가지표

- Impact Factor(IF)는 **저널**의 영향력을 수치화한 것 (논문의 영향력 X)

$$\text{2016 Impact Factor} = \frac{\text{해당 저널에 수록된 2014, 2015년 논문을 2016년 다른 논문에서 인용한 횟수}}{\text{2014, 2015년에 해당 저널에 수록된 논문 수}}$$

- 분모 : 순수 연구 논문과 리뷰 논문만이 분모의 계산에 사용됨. 레터, 사설 등은 제외
- 분자 : 모든 문헌 유형에 대한 인용 횟수가 반영됨

- A 저널의 2016 IF 계산방법

2014년 논문 수  
Published in 2014  
**15**

2014년 논문 인용 횟수(2016년 기준)  
Cited in 2016 from published in 2014  
**9**

2015년 논문 수  
Published in 2015  
**27**

2015년 논문 인용 횟수(2016년 기준)  
Cited in 2016 from published in 2015  
**18**

2016 A저널의 Impact Factor

$$\frac{9+18}{15+27} = 0.628$$

## • 저널랭킹

- 충북대학교 중앙도서관 > 전자자료  
> 저널평가정보

### - JCR(Journal Citation Reports)

SCIE, SSCI, A&HCI 의 인용색인 데이터를 바탕으로 각 저널에 대한 인용통계정보를 제공하는 저널평가 데이터베이스

### - KCI(Korea Citation Index)

한국연구재단에서 운영하는 한국판 SCI

소장자료검색

전자자료

도서관 서비스

전자자료통합검색

저널검색(AtoZ)

Web DB/전자저널

국내Web DB/전자저널

국외Web DB/전자저널

E-Book/Audio Book

E-Learning

주제가이드

저널평가정보

연구정보관리도구(RefWorks)

교외접속(Proxy)

협력기관 자료이용

전자자료  
e-References

충북대학교 도서관에서 제공

중앙도서관 학술

저널랭킹

마스터 저널 리스트

연구자

- JCR 랭킹

- SJR 랭킹

- CiteScore 랭킹

- KCI 랭킹

- JCR 주제별 요약

- SJR 주제별 요약

- CiteScore 주제별 요약

- KCI 주제별 요약

- SCI(E)/SSCI/A&HCI

- ESCI/CC

- SCOPUS

- MEDLINE

- DOAJ

- KCI

- EMBASE

- Beall's List

- WASET & OMICS List

- 논문 출판 동향

- 저널 선택 서비스

- 관심저널관리

- 관심저널비교



# 선정 논문의 정보

- 선정 논문의 정보 확인

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9093073>

Published in: [IEEE Transactions on Semiconductor Manufacturing](#) ( Volume: 33, [Issue: 3](#), Aug. 2020)

- IF (SCI) 와 CiteScore (Scopus) 의 차이

- IF는 산정기간이 2년이나, **CiteScore**는 산정기간이 3년.

즉, 지난 3년간 한 아이템이 받은 평균 인용횟수로 영향력을 계산

- **Scopus**에 색인된 2만 2천 여종의 저널을 대상으로 합니다.

- 출판된 논문(article)에 국한되지 않고, 뉴스, 사설, 레터 등을 포함하여 잠재적으로 인용 가능한 모든 문서를 기초로 합니다.

- 구글 스칼라(scholar) : 확장 프로그램

A Deep Convolutional Neural Network for Wafer Defect Identification on an Imbalanced Dataset in Semiconductor Manufacturing Processes

- 충북대 중앙도서관

# [PR] 초록(Abstract)

## 요약

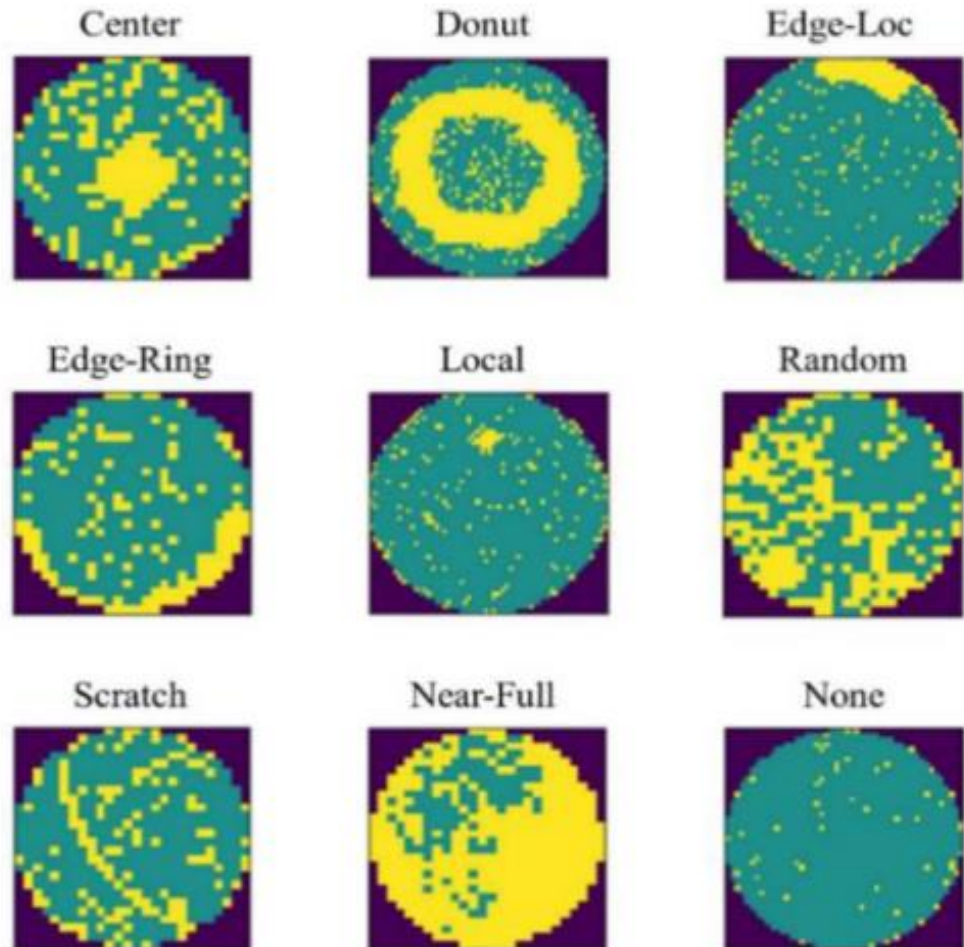
웨이퍼 맵에는 웨이퍼 표면의 다양한 불량 패턴에 대한 정보가 포함되어 있으며 이러한 **불량의 자동 분류**는 근본 원인을 찾는 데 중요한 역할을 합니다. 반도체 엔지니어는 웨이퍼 불량 분류를 위해 수동 육안 검사 또는 수동으로 유용한 특징을 추출하는 기계학습 기반 알고리즘과 같은 다양한 방법을 적용합니다. 그러나 이러한 방법은 신뢰할 수 없으며 분류 성능도 좋지 않습니다. 따라서 **본 논문에서는 자동 웨이퍼 불량 식별을 위한 딥러닝 기반 컨볼루션 신경망(CNN-WDI)을 제안합니다.** 본 논문에서는 클래스 불균형 문제를 극복하기 위해 **데이터 증량기**를 적용했습니다. 제안된 모델은 수동으로 특징을 추출하는 대신 컨볼루션 계층을 사용하여 가치 있는 특징을 추출합니다. 또한 CNN-WDI 모델의 분류 성능을 향상시키기 위해 **배치 정규화(Batch Normalization) 및 공간 드롭아웃(Spatial Dropout)과 같은 최신 규제화(regularization) 방법을** 사용합니다. 실제 웨이퍼 데이터셋을 사용한 실험 결과 비교는 우리 모델이 이전에 제안된 모든 기계학습 기반 웨이퍼 불량 분류 모델보다 성능이 우수함을 보여줍니다. 9개의 서로 다른 웨이퍼 맵 불량이 있는 CNN-WDI 모델의 평균 분류 정확도는 96.2%로, 동일한 데이터셋을 사용한 **이전의 최고 평균 정확도에서 6.4% 증가**했습니다.

1. 배경
  - 정의
  - 중요성
2. 문제점
  - 저성능
3. 제안(해법)
  - CNN
  - 증량
  - 자동 특징추출
  - 배치정규화
  - 드롭아웃
4. 결과/토의
  - 우수성(비교)
  - 기여

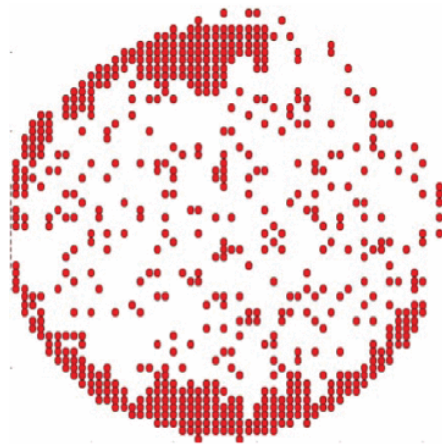


# 1. 서론(Introduction)

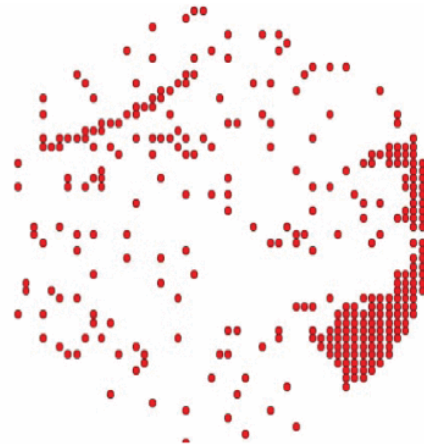
- 고품질 IC를 생산하기 위해서는 웨이퍼가 아주 깨끗하고 고도로 정렬되어야 함
- 고도로 자동화되고 정밀한 장비, 숙련된 엔지니어도 웨이퍼 불량생산 불가피
- 웨이퍼 검사 → 2차원 웨이퍼 맵(WM, wafer map) 생성 → 불량 패턴을 분류
  - WM 불량 분석은 반도체 제조의 비정상 프로세스를 발견하고 조치를 취하는데 정보를 제공하므로 중요함
  - 불량패턴: 9종



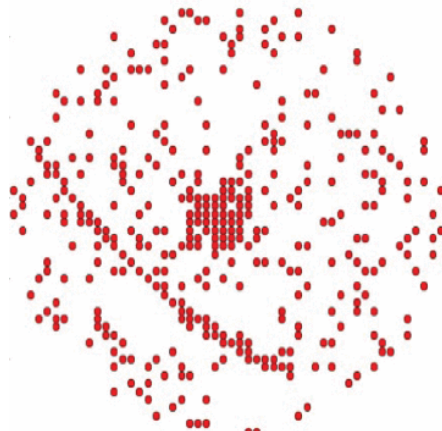
# [참고] 불량패턴의 혼합



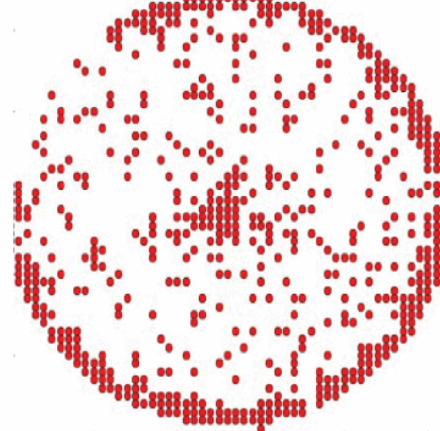
(a) Ring and zone



(b) Scratch and zone



(c) Circle and scratch



(d) Circle and ring

K. Kyeong and H. Kim, "[Classification of Mixed-Type Defect Patterns in Wafer Bin Maps Using Convolutional Neural Networks](#)," in *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 3, pp. 395-402, Aug. 2018, doi: 10.1109/TSM.2018.2841416.

# 1. 서론(Introduction)

- 기존의 웨이퍼 맵 검사방법
  - 숙련된 공정 엔지니어가 고해상도 현미경으로 **수동 검사**
  - 머신러닝 기반 분류 시스템 : **수동 특징 추출** + 분류기(SVM, DT, 앙상블 모델)
  - 딥러닝 기반 CNN : **자동 특징 추출** + 분류기

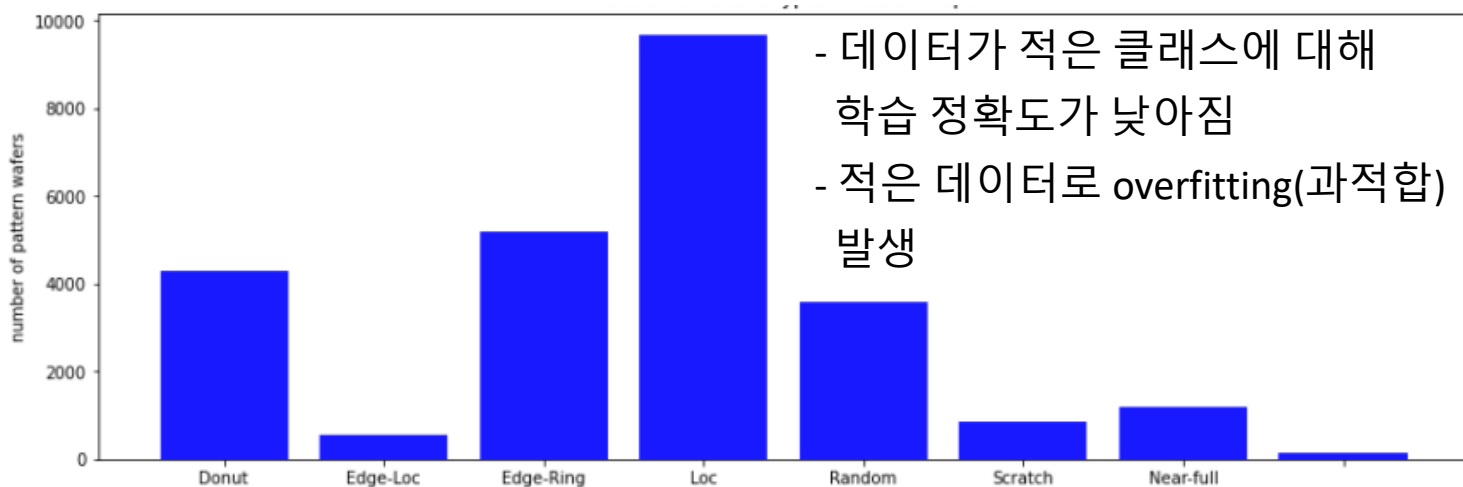
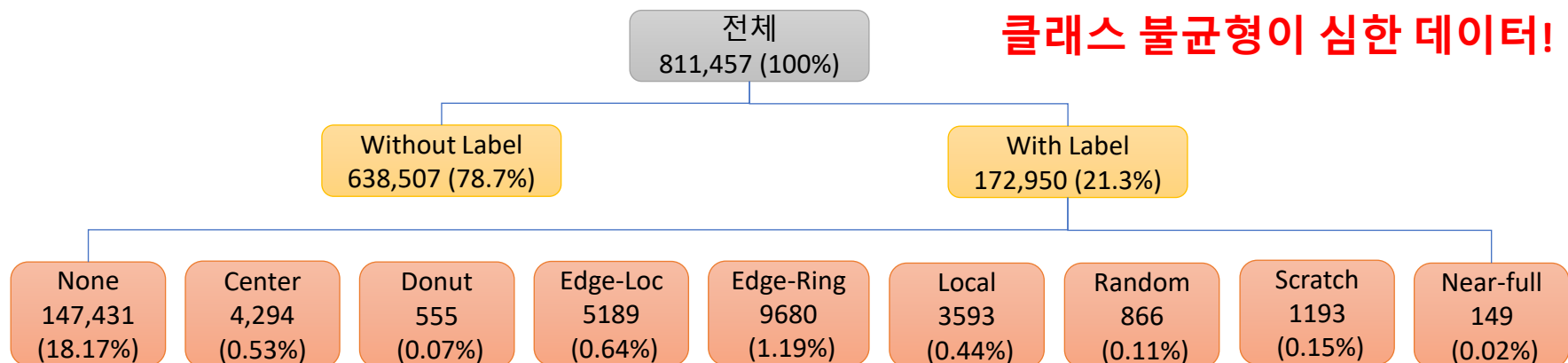
But **데이터 불균형 문제**로 클래스별 정확도 편중현상 (소수 클래스 부정확)
- 본 논문에서 제안하는 방법
  - 딥러닝 기반 CNN + **데이터 증량(augmentation) 기법** → 평균 정확도 96.2%  
(기존 최고성능 대비 6.4% 상승)

## 2. 방법(Methodology)

Kaggle: <https://www.kaggle.com/qingyi/wm811k-wafer-map/code>

### A. 데이터셋

- 반도체 제조공정 46,293개 로트에서 WM 총 811,457개 수집
- 해상도 :  $(6 \times 21) \sim (300 \times 202)$  범위의 총 632개의 다양한 크기

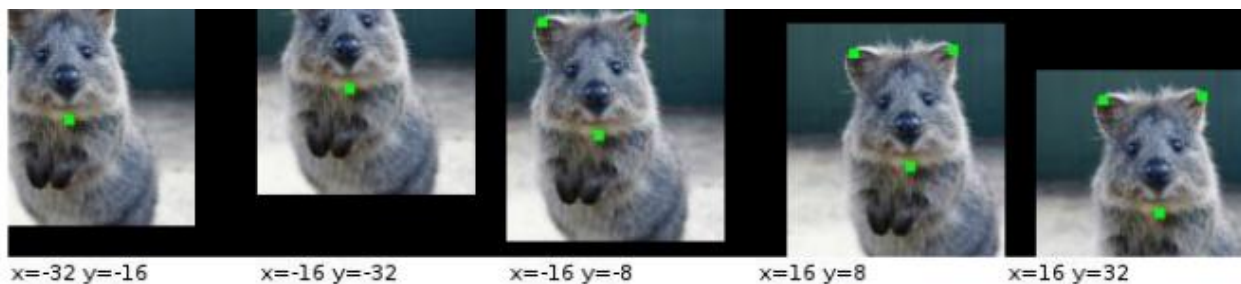


## 2. 방법(Methodology)

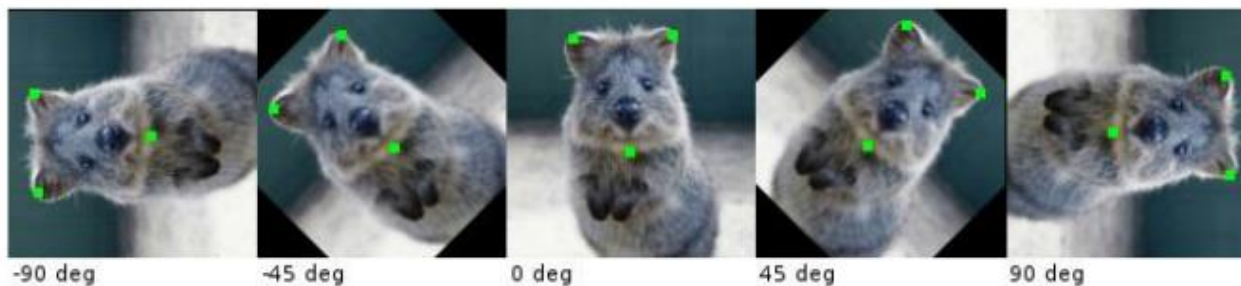
### B. 데이터 증량

- 원본 데이터를 무작위로 늘림으로써 데이터셋의 다양성과 크기를 모두 향상
- 일반적인 증량 방법 : 수평/수직 대칭, 수평/수직 이동, 회전, 확대/축소  
→ 불량크기의 크기, 위치, 방향 변화에 대한 내성을 갖도록 함

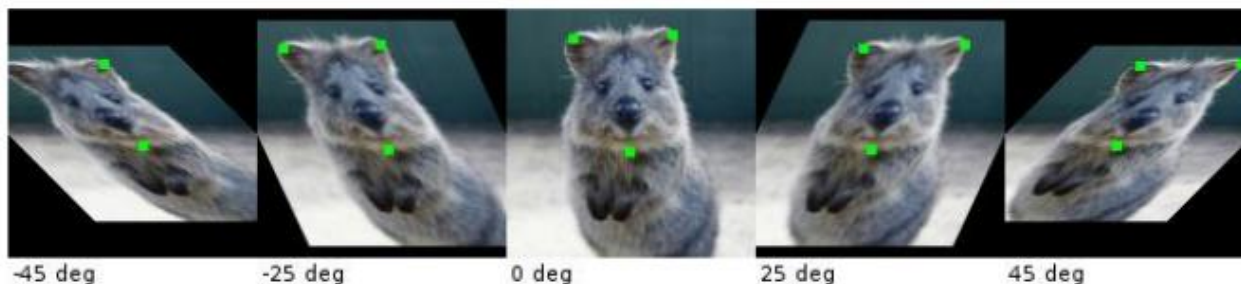
Translate



Rotate



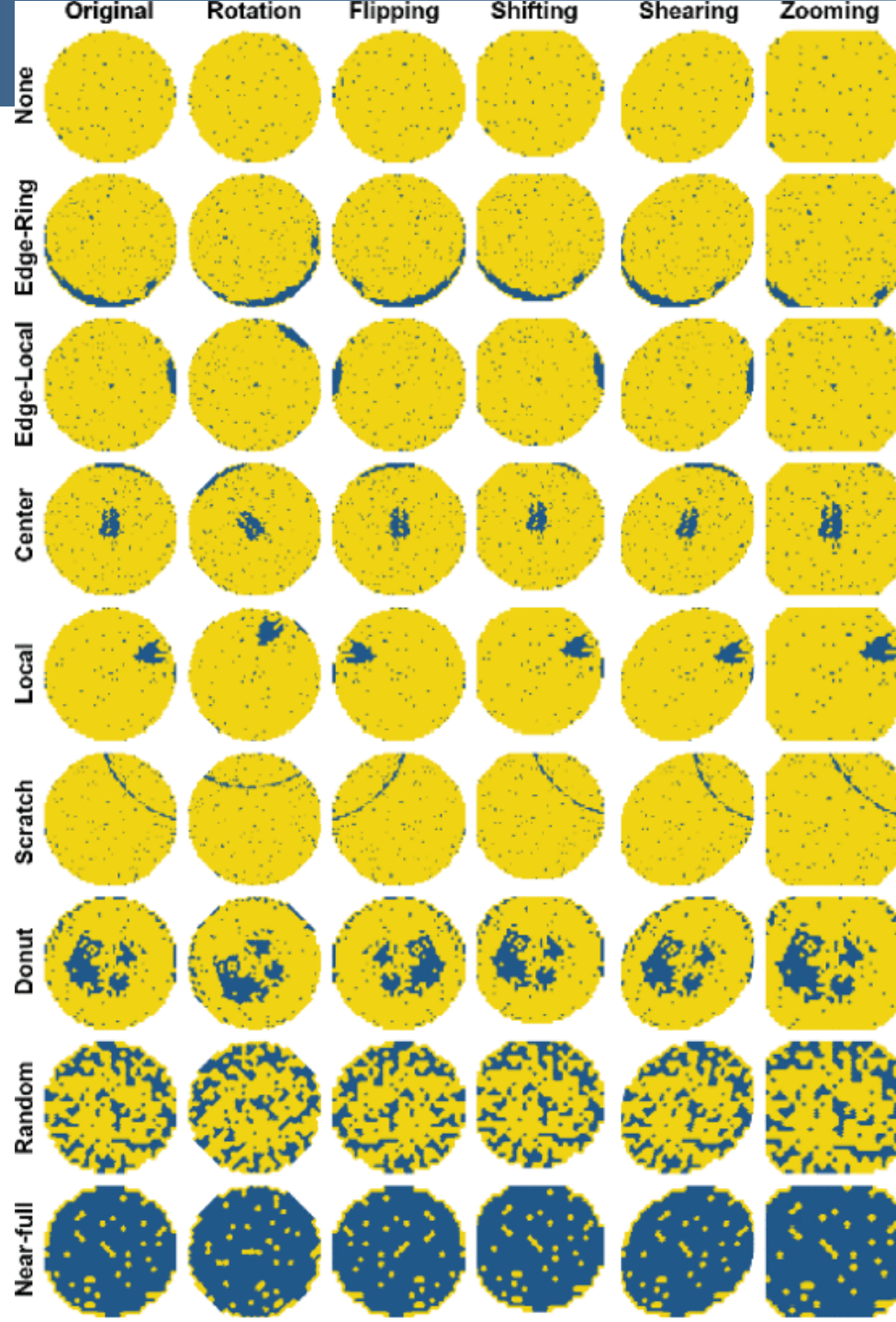
Shear





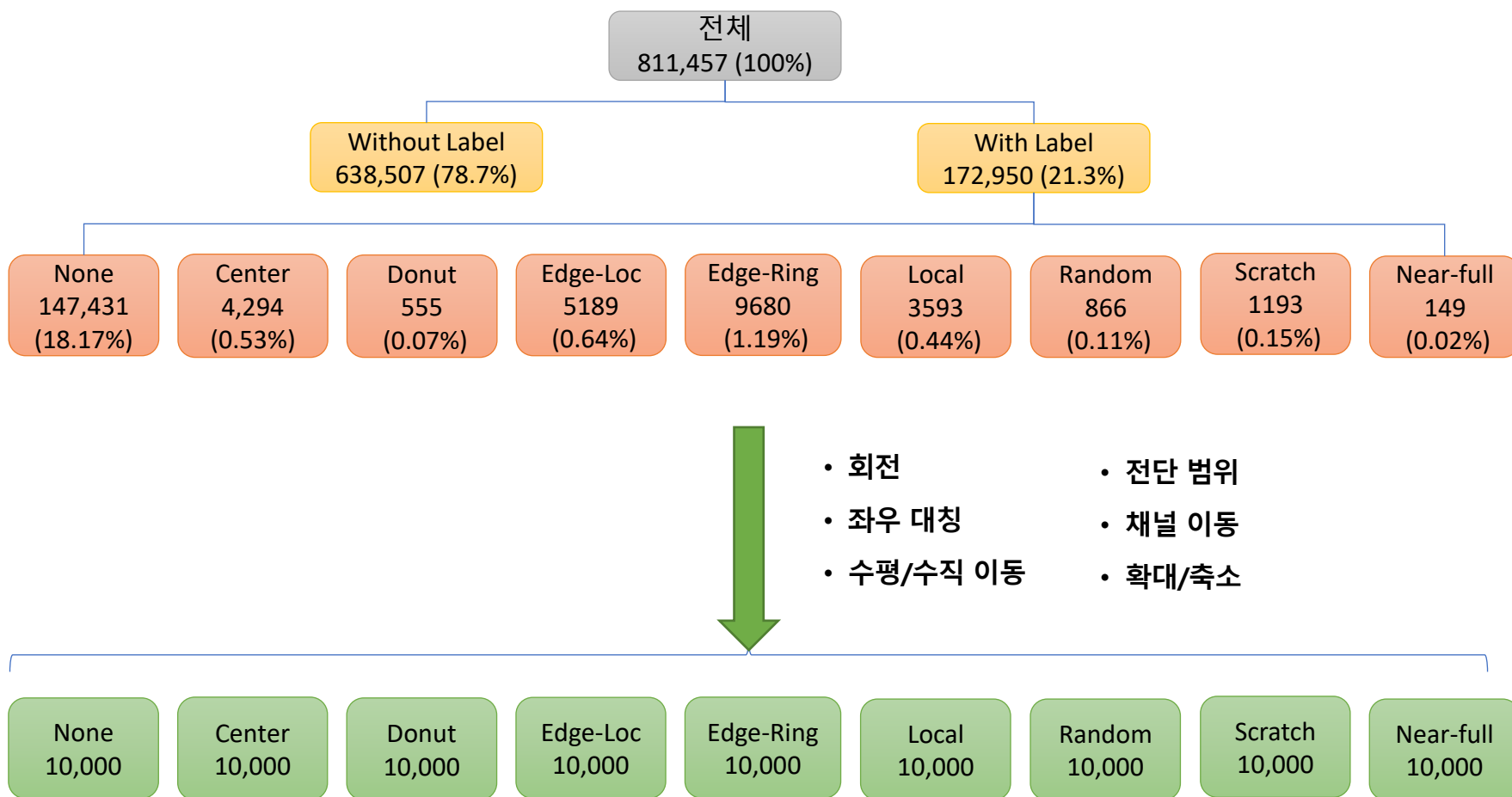
# [참고] 불량패턴의 생성

H. S. Shon, E. Batbaatar, W. -S. Cho and S. G. Choi, "[Unsupervised Pre-Training of Imbalanced Data for Identification of Wafer Map Defect Patterns](#)," in IEEE Access, vol. 9, pp. 52352-52363, 2021, doi: 10.1109/ACCESS.2021.3068378.



## 2. 방법(Methodology)

### c. 데이터 전처리 → 균형잡힌 데이터셋 생성

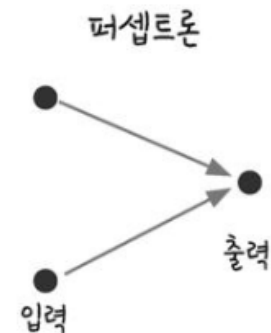
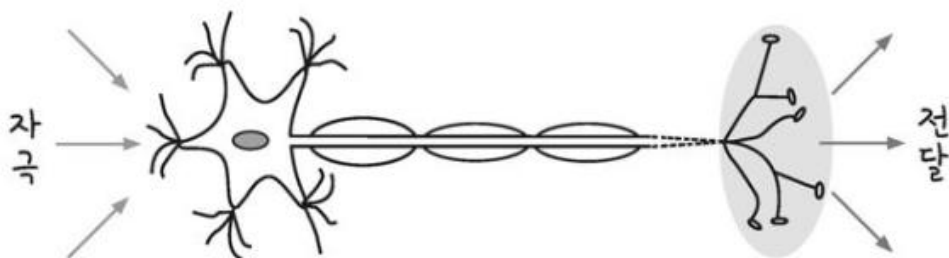
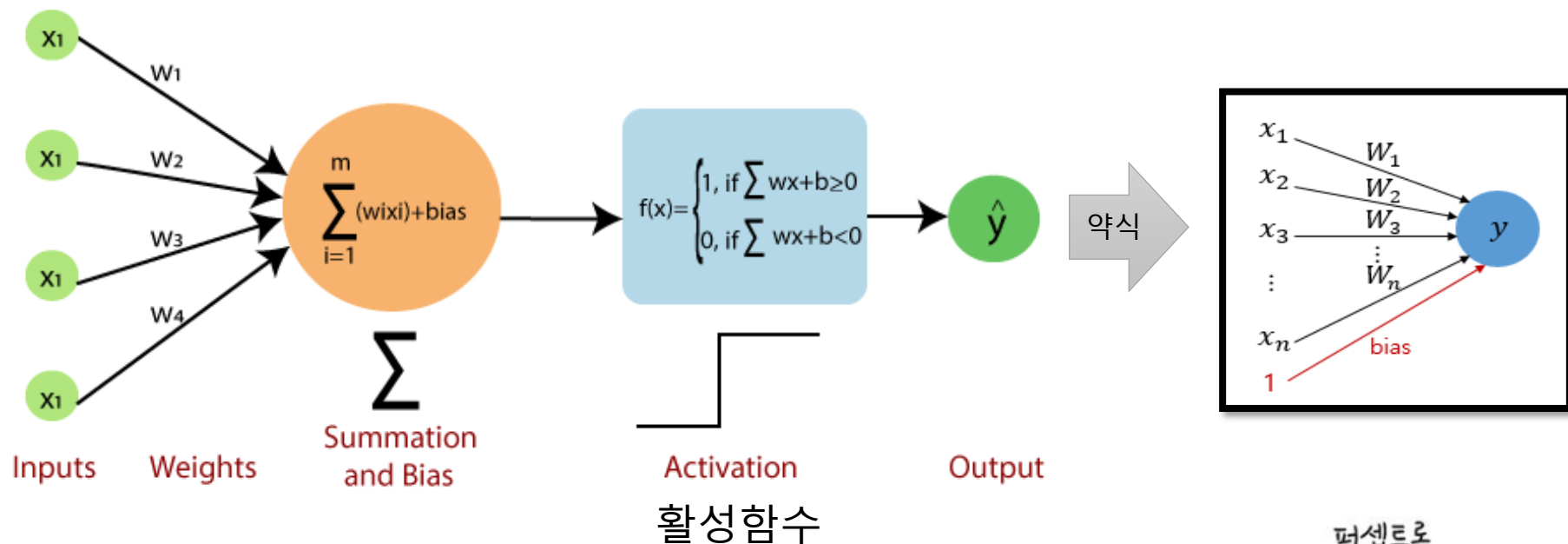


Train : Validation : Test = 65 : 20 : 15

# [참고] Perceptron

- **Perceptron = 1개의 Neural Network (신경망)**

- 로젠블라트가 1957년 제안한 인간 두뇌에 대한 수학적 모델

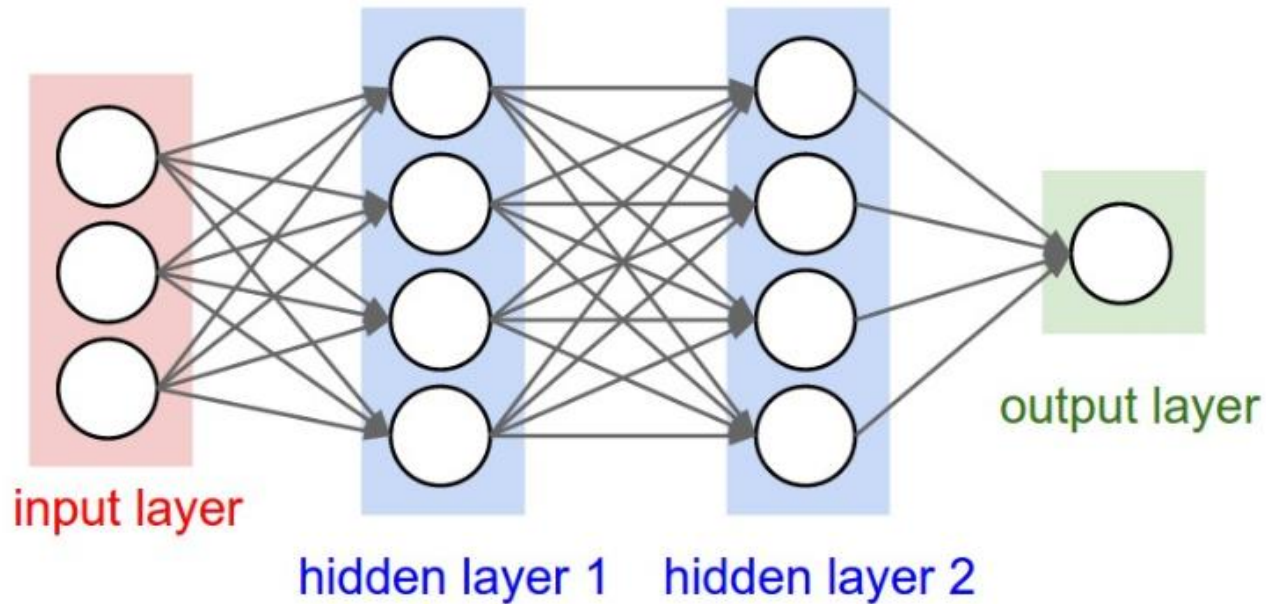





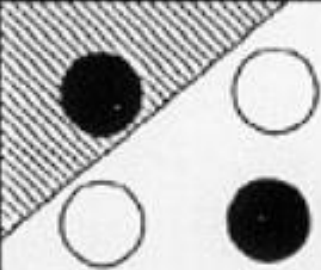

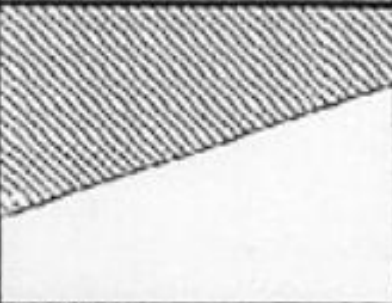
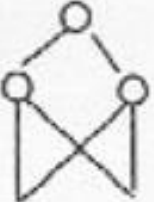
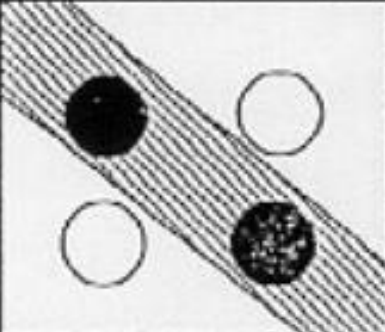

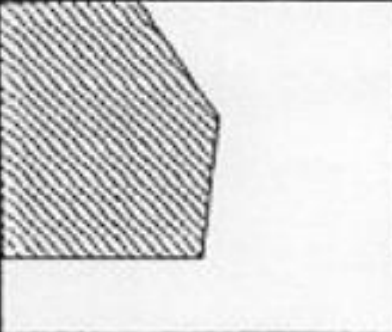
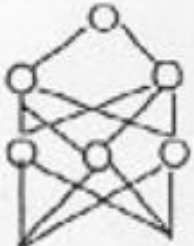
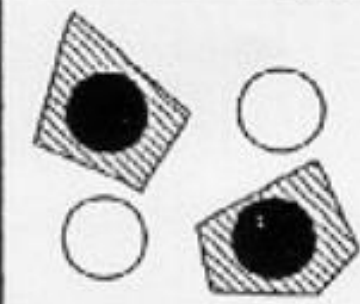


# [참고] 다층 퍼셉트론

- 다층 퍼셉트론 (MLP, Multi-Layer Perceptron)

- 은닉층 1~2개인 신경망

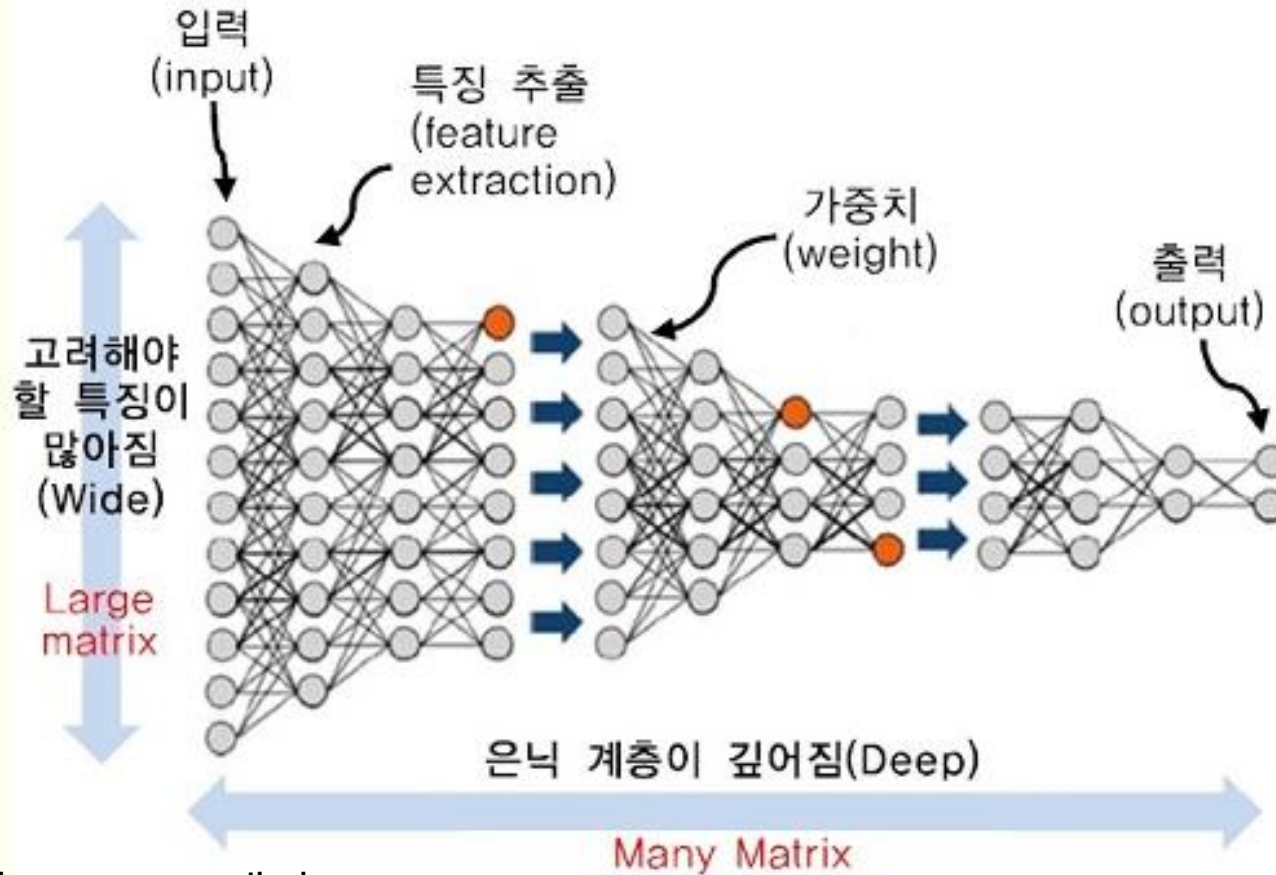


# [참고] 신경망의 깊이(은닉층)에 따른 표현력

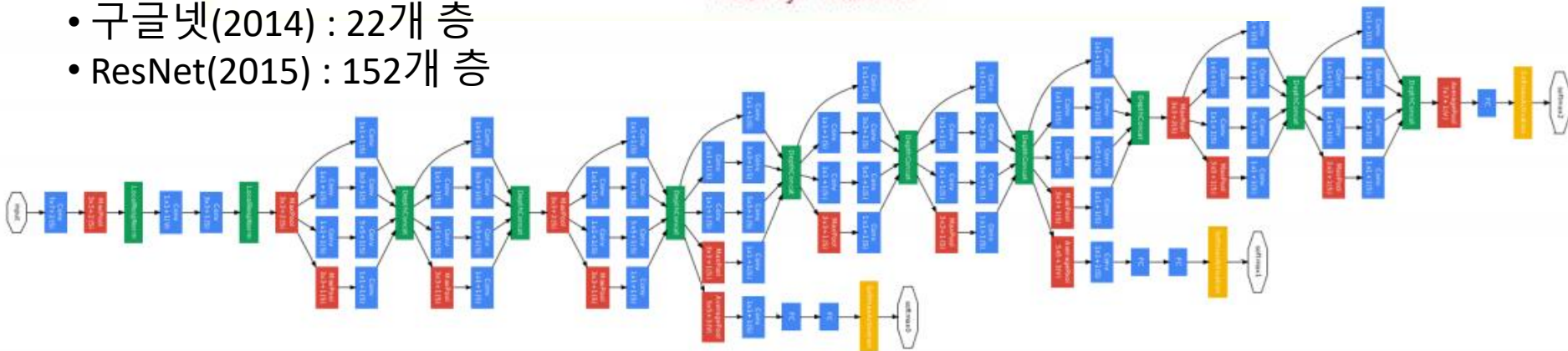
Structure	Description of decision regions	Exclusive-OR problem	Classes with meshed regions	General region shapes
 Single layer	Half plane bounded by hyperplane			
 Two layer	Arbitrary (complexity limited by number of hidden units)			
 Three layer	Arbitrary (complexity limited by number of hidden units)			

은닉층이 많을수록 결정영역의 표현력(정확도)가 높아짐  
 그러나, 은닉층 2~3개 이상인 경우 **학습이 불가능** → 신경망 암흑기, 통계기법 발전

# [참고] 심층 신경망(DNN, Deep Neural Network)



- 구글넷(2014) : 22개 층
- ResNet(2015) : 152개 층

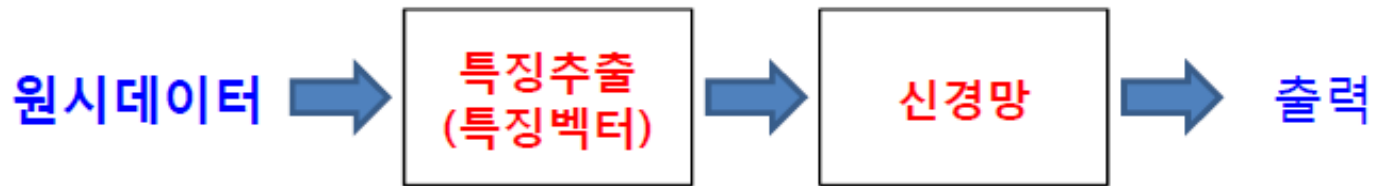


# [참고] 심층 신경망(DNN, Deep Neural Network)

## ❖ 일반 신경망과 심층 신경망

### ▪ 일반 신경망 모델

- 원시 데이터(original data)에서 직접 특징(handcrafted feature)을 추출해서 만든 **특징 벡터(feature vector)**를 입력으로 사용
- 특징 벡터들의 품질에 영향



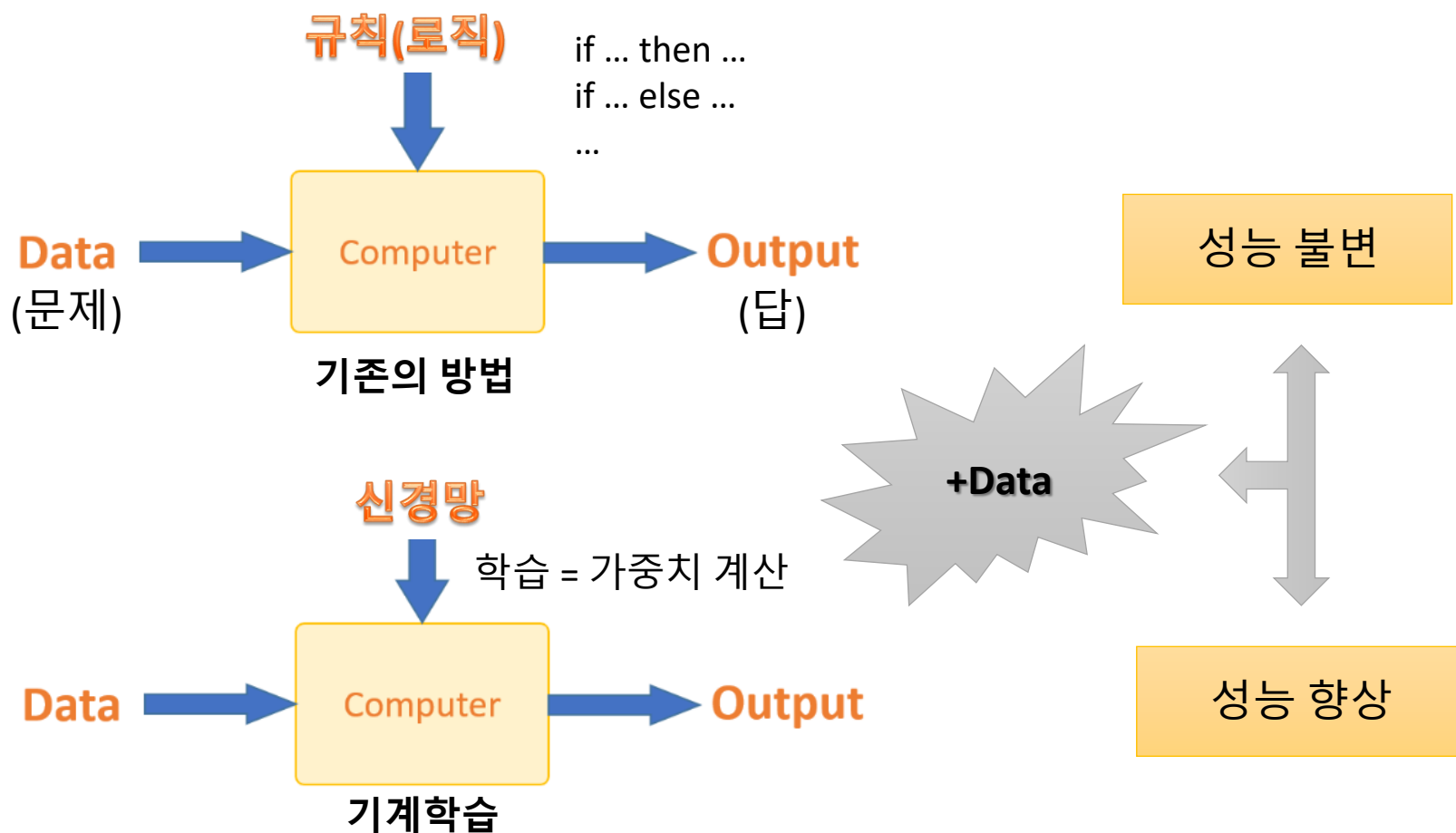
### ▪ 심층 신경망

- 특징추출과 학습을 함께 수행
- 데이터로부터 **효과적인 특징**을 학습을 통해 추출 → 우수한 성능



# [참고] 규칙(rule) vs 학습(learning)

- 기계학습은 명시적인 프로그래밍 없이 컴퓨터가 스스로 학습하는 것(사무엘, 1959)
- 기계가 데이터(경험)가 증가함에 따라 성능이 향상되었다면,  
그 기계는 데이터(경험)를 통해 '학습'한 것이다. (미첼, 1997)

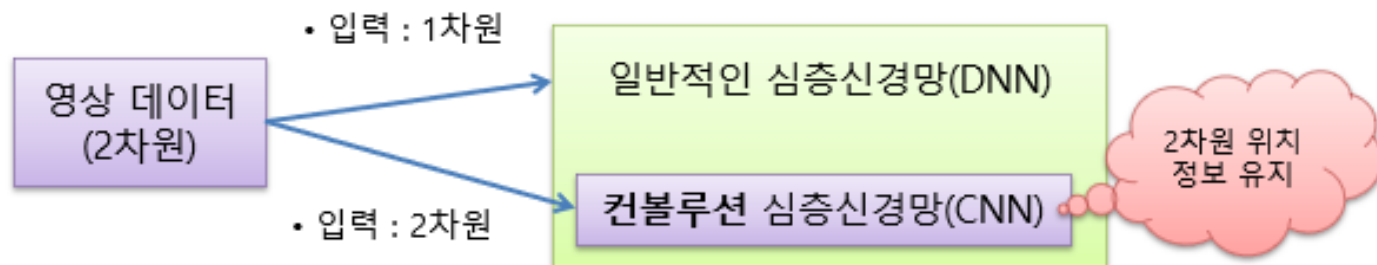




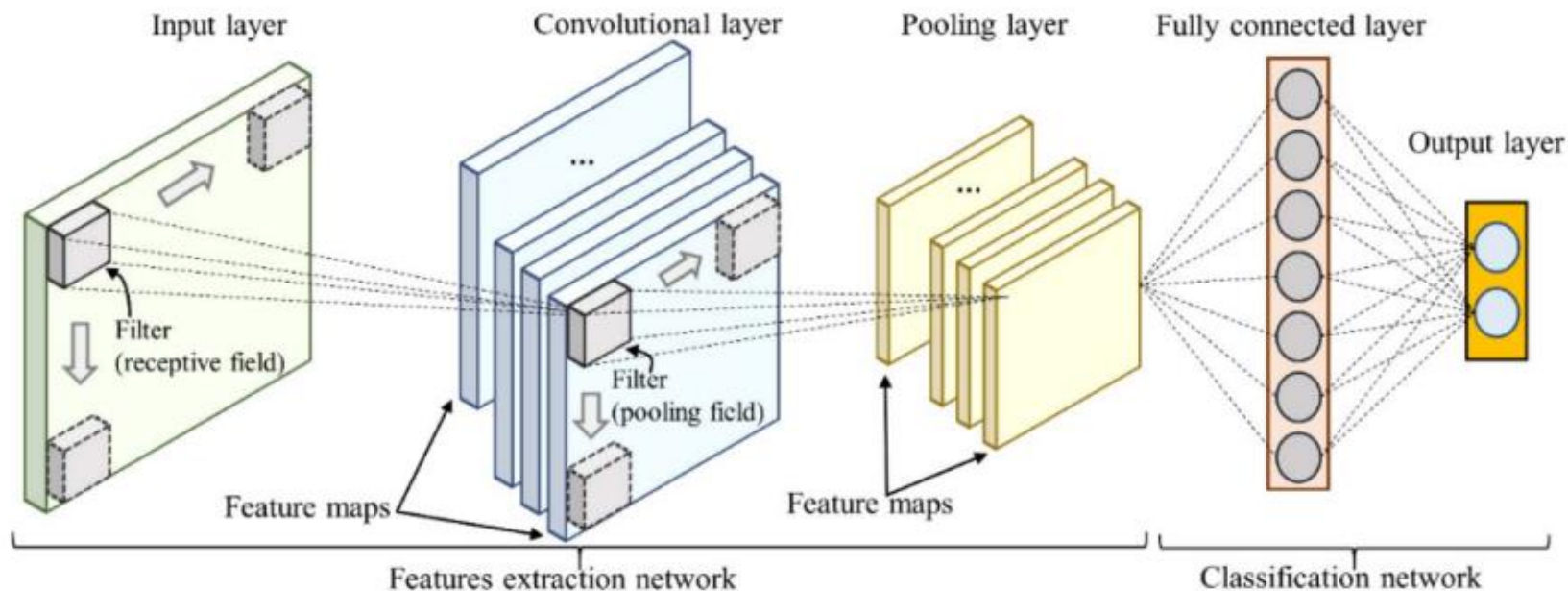
## 2. 방법(Methodology)

### D. 컨볼루션 신경망(CNN)

- MLP의 발전된 형태이며 이미지 처리에 적합한 신경망
  - 인간의 시각 피질과 유사하고 2차원(2D) 특징을 추출하고 학습
  - 같은 크기의 DNN에 비해 파라미터 수가 매우 적다.



- CNN의 구조



# [참고] CNN 구조 – 컨볼루션(Convolution) 층

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

298

+

-491

+

487

+ 1 = 295

Bias = 1

입력

필터

(학습)

특징맵

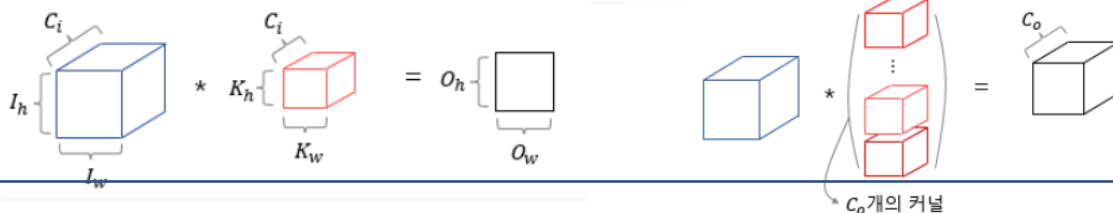
Output

-25	466	466	475	...
295				...
				...
				...
...	...	...	...	...

• 위 연산에서 사용되는 커널은 3개의 채널을 가지는 1개의 커널

• 가중치 매개변수의 총 수 :  $\{(K_h * K_w) * C_i + 1\} * C_o$

=  $\{(커널 크기) * 입력채널 + 1\} * 출력채널$



# [참고] CNN 구조 – 컨볼루션층

- **Stride(보폭)** : 필터를 이동시키는 간격
- **Padding** : 이미지의 가장자리에 데이터(zero 등)를 추가하여  
컨볼루션 이후에도 이미지의 사이즈가 줄지 않도록 함

1	2	2	0	9	1
2	3	5	0	9	2
3	4	7	8	9	3
4	5	8	9	9	9
4	5	8	3	9	3
2	1	2	2	9	2

< Stride = 1 >

1	2	2	0	9	1
2	3	5	0	9	2
3	4	7	8	9	3
4	5	8	9	9	9
4	5	8	3	9	3
2	1	2	2	9	2

< Stride = 2 >

0	0	0	0	0	0
0	7	1	1	1	0
0	6	6	6	1	0
0	1	5	5	5	0
0	1	1	4	4	0
0	0	0	0	0	0

< 0 padding >



# [참고] CNN 구조 – 풀링(Pooling) 층

CNN에서 Pooling은 통해 데이터의 크기를 줄이고 특징을 추출해 낸다.

## • Pooling(Subsampling)

- Pooling 계층은 주위의 픽셀을 묶어서 하나의 대표 픽셀로 바꾼다. 즉 이미지의 차원을 축소한다.
- 이미지의 인식 대상이 한 쪽으로 치우치거나 돌아가 있어도 어느 정도 보상을 해 줌.
- 이미지 크기를 줄이고 과적합을 방지하는 데 도움이 됨

Max Pooling (최대 풀링)	Average Pooling (평균 풀링)	Stochastic Pooling (확률적 풀링)
---------------------	-------------------------	-----------------------------

- local의 정보를 추출

5	-5
-5	5

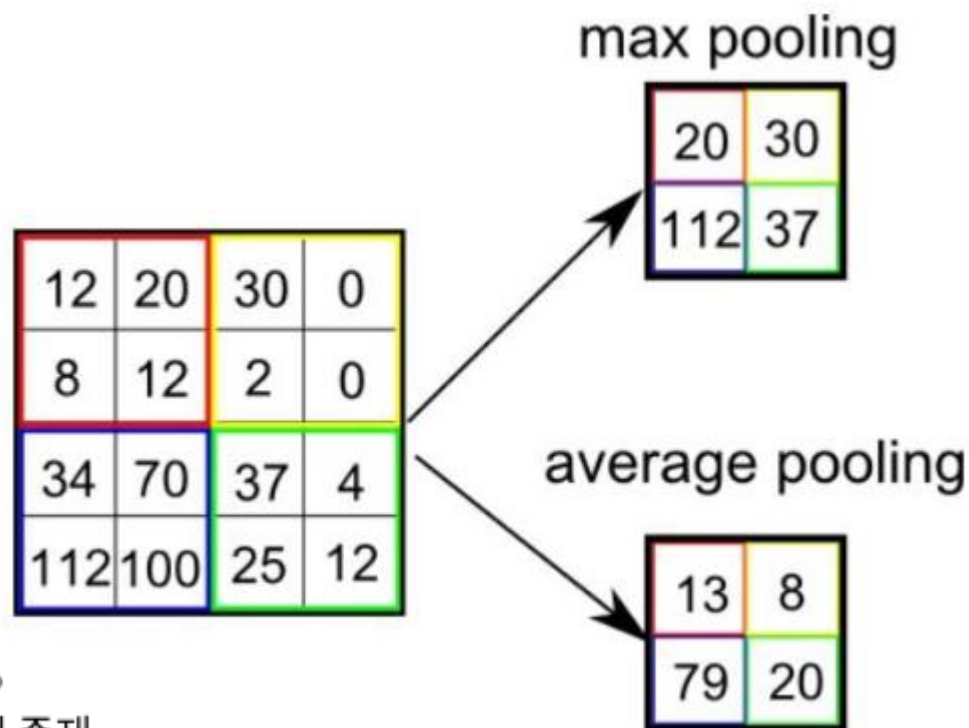
0
---

Stride = 2

Pooling 후 값이 높다는 것의 의미?

‘어떤 형태의 representation이 그 영역 안에 존재

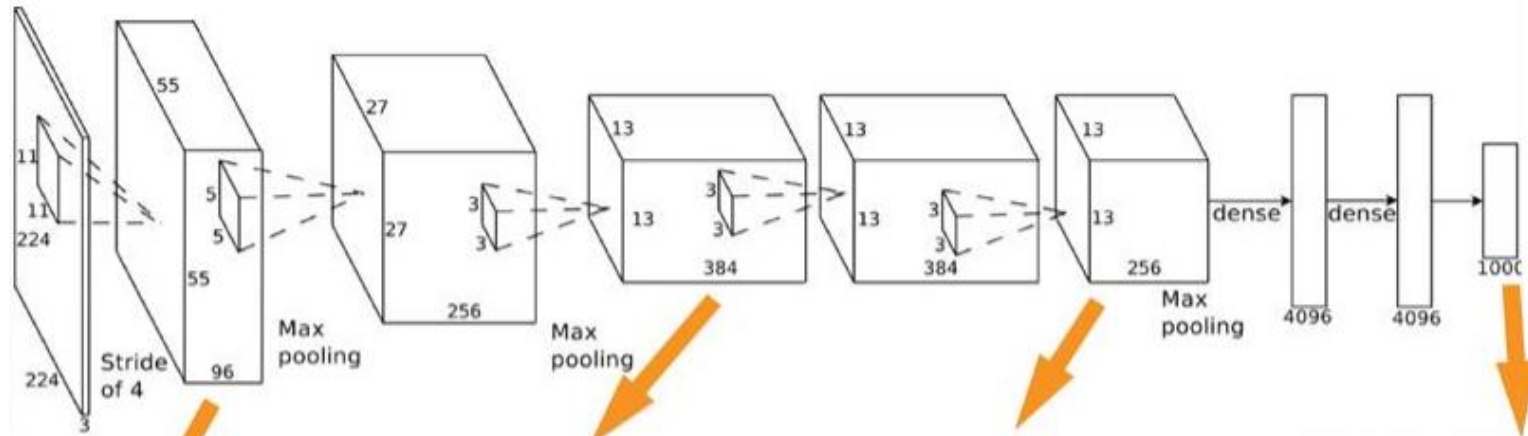
‘이 경우에는, 대각선 성분이 존재!’



# [참고] CNN의 깊이와 추상화



깊어질수록 보다 고차원적인, 추상화된 속성을 얻음



Conv 1: Edge+Blob



Conv 3: Texture



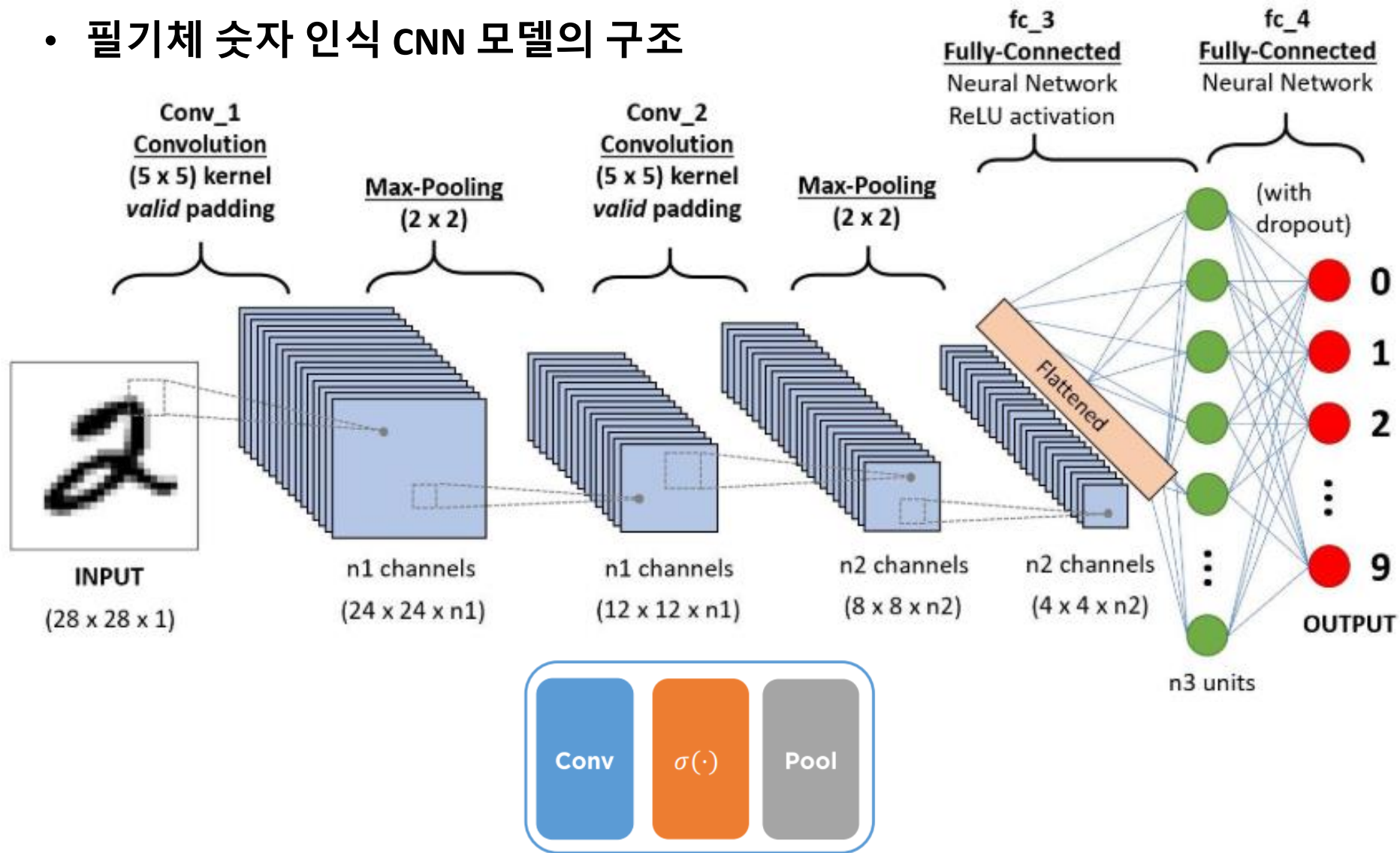
Conv 5: Object Parts



Fc8: Object Classes

# [참고] CNN 구조 – 예시 : LeNet

## • 필기체 숫자 인식 CNN 모델의 구조



Convolution block

활성함수  $\sigma(\cdot)$ 를 가하지 않으면? Linear regression ☺

# [참고] DNN과 CNN의 가중치 개수

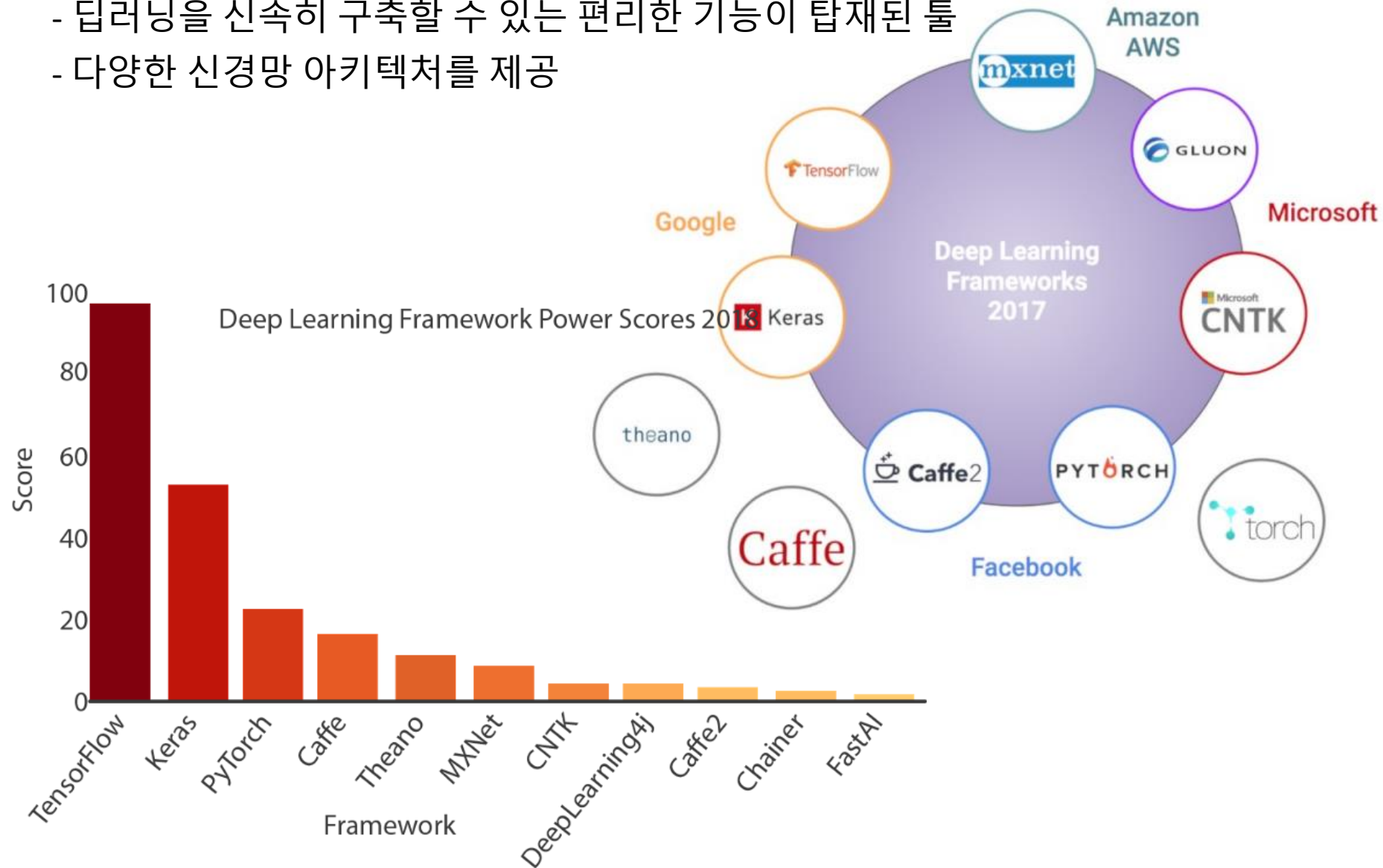
- AlexNet : Conv 6층 + FC 2층 → [총가중치] Conv + Pool : **3,747,200개** → GPU 사용  
MLP : **58,621,952개**

layers	필터/블록 크기	필터 개수	스트라이드	패딩	노드개수 (출력 크기)	학습대상 가중치 개수
input					224x224x3 (=150,528)	
Conv:1	11x11x3	96	4	3	55x55x96 (=290,400)	(11x11x3+1)x96 (=34,944)
Pool:1	3x3		2		27x27x96 (=69,984)	
Conv:2	5x5x96	256	1	2	27x27x256 (=186,624)	(5x5x96+1)x256 (=614,656)
Pool:2	3x3		2		13x13x256 (=43,264)	
Conv:3	3x3x256	384	1	1	13x13x384 (=64,896)	(3x3x256+1)x384 (=885,120)
Conv:4	3x3x384	384	1	1	13x13x384 (=64,896)	(3x3x384+1)x384 (=1,327,488)
Conv:5	3x3x384	256	1	1	13x13x256 (=43,264)	(3x3x384+1)x256 (=884,992)
Pool:5	3x3		2		6x6x256 (=9,216)	
FC:6					4096	6x6x256x4096 (=37,748,736)
FC:7					4096	4096x4096 (=16,777,216)
FC:8					1000	4096x1000 (=4,096,000)

# [참고] 딥러닝 프레임워크(Framework)

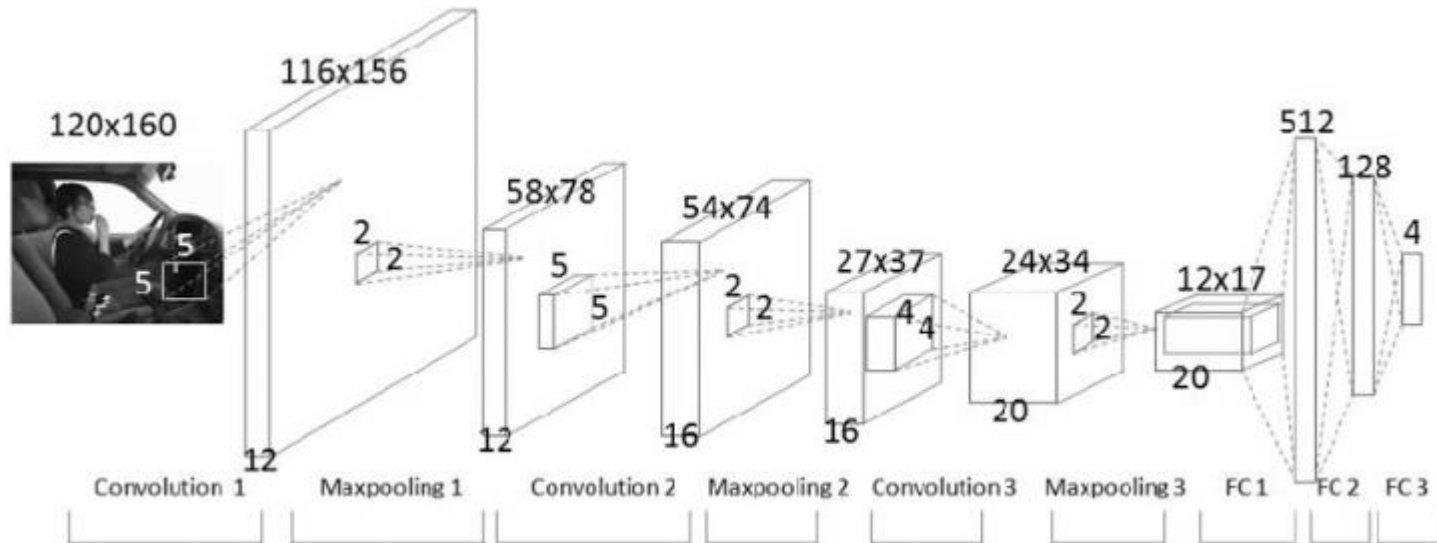
- 딥러닝 프레임워크란?

- 딥러닝을 신속히 구축할 수 있는 편리한 기능이 탑재된 툴
- 다양한 신경망 아키텍처를 제공





# [참고] CNN 구현 – Keras



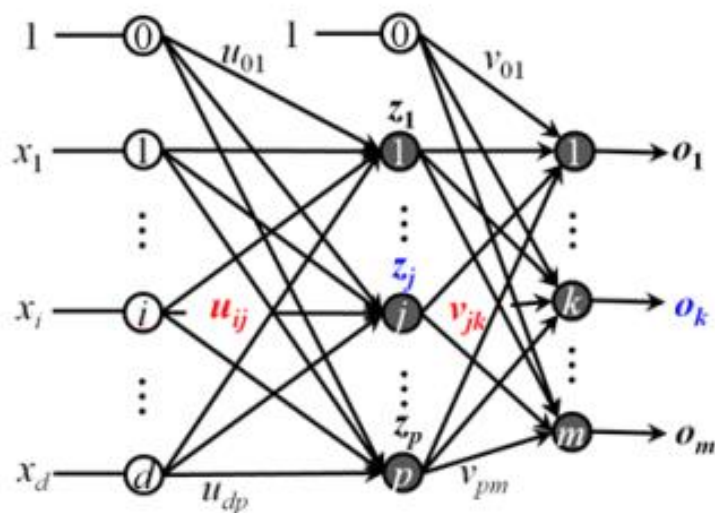
```
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten

model = Sequential()
model.add(Conv2D(12, kernel_size=(5, 5), activation='relu', input_shape=(120, 160, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(20, kernel_size=(4, 4), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

# [참고] 학습(Learning) – 오차 역전파 알고리즘

## ❖ 다층 퍼셉트론 MLP의 학습

- 오차 역전파 알고리즘(Error back propagation algorithm, Backprop algorithm)



$$osum_k = \sum_{j=1}^p v_{jk} z_j + v_{0k} \quad (1 \leq k \leq m)$$

$$o_k = f(osum_k)$$

$$zsum_j = \sum_{i=1}^d u_{ij} x_i + u_{0j} \quad (1 \leq j \leq p)$$

$$z_j = f(zsum_j)$$

입력  $(x_1, x_2, \dots, x_d)$

출력  $(y_1, y_2, \dots, y_m)$

$$E = \frac{1}{2} \sum_{k=1}^m (o_k - y_k)^2 \quad \text{오차함수}$$

$$\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \eta \frac{\partial E}{\partial \mathbf{v}}$$

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \eta \frac{\partial E}{\partial \mathbf{u}}$$

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial v_{jk}} = (o_k - y_k) f'(osum_k) z_j = \delta_k z_j$$

$$\frac{\partial E}{\partial u_{ij}} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial u_{ij}} = \sum_{k=1}^m \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial z_j} f'(zsum_j) x_i$$

$$= \sum_{k=1}^m (o_k - y_k) f'(osum_k) v_{jk} f'(zsum_j) x_i$$

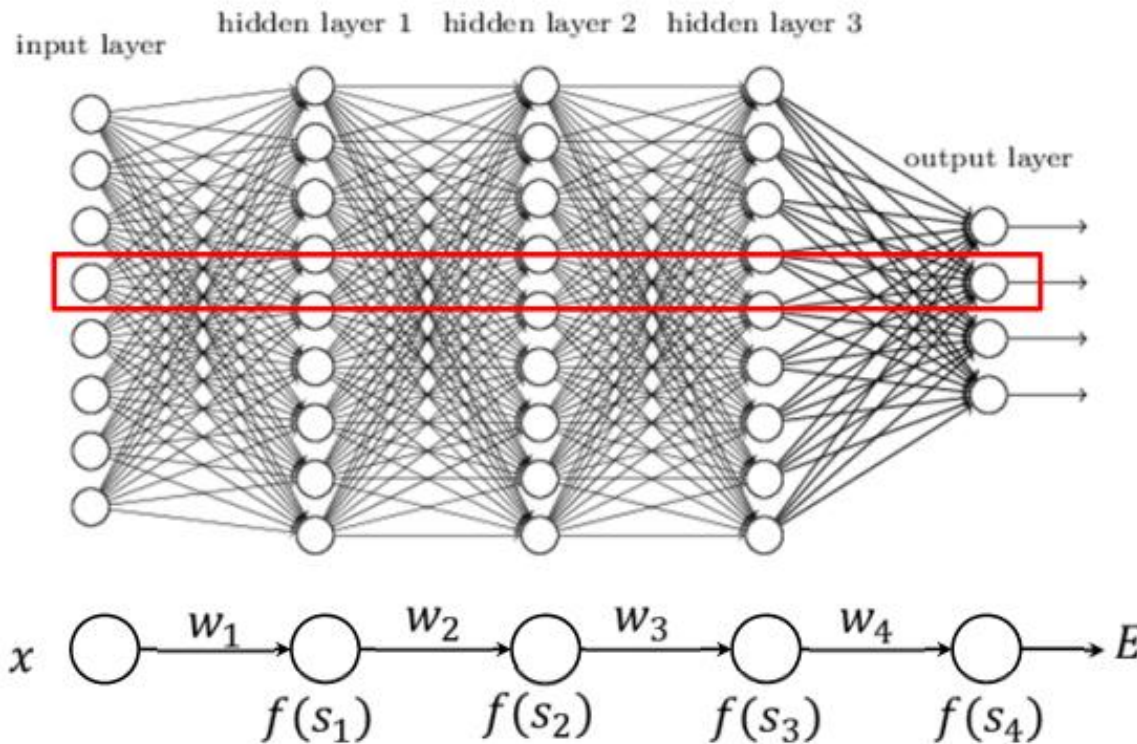
$$= \sum_{k=1}^m \delta_k v_{jk} f'(zsum_j) x_i$$

# [참고] 학습(Learning) – 오차 역전파 알고리즘

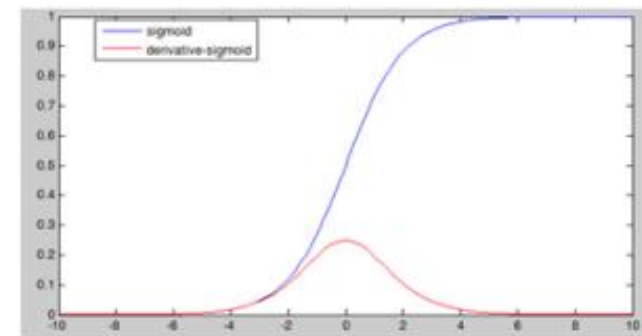
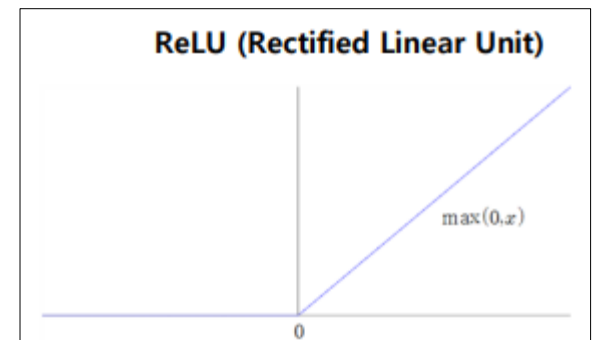
## ❖ 심층 신경망 학습의 어려움

### ▪ 기울기 소멸 문제(vanishing gradient problem)

- 은닉층이 많은 다층 퍼셉트론에서, 출력층에서 아래 층으로 갈 수록 전달되는 오차가 크게 줄어들어, 학습이 되지 않는 현상



$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f(s_4)} f'(s_4) w_4 f'(s_3) w_3 f'(s_2) w_2 f'(s_1) x$$





# [참고] 학습(Learning) – 오차 역전파 알고리즘

## ❖ 심층 신경망 학습 전략

### ▪ 사전 학습(pre-training)

- 가중치를 무작위로 초기화하는 대신
- 데이터를 사용하여 비지도학습에 의한 초기 가중치 결정

### ▪ 층(layer)간 노드 연결 수 제한

- 직전 층의 모든 노드와 연결하는 대신
- 일부 지역적 노드와 연결

### ▪ 가중치 공유

- 같은 층의 여러 노드가 동일한 가중치 사용

### ▪ 전달함수 변경

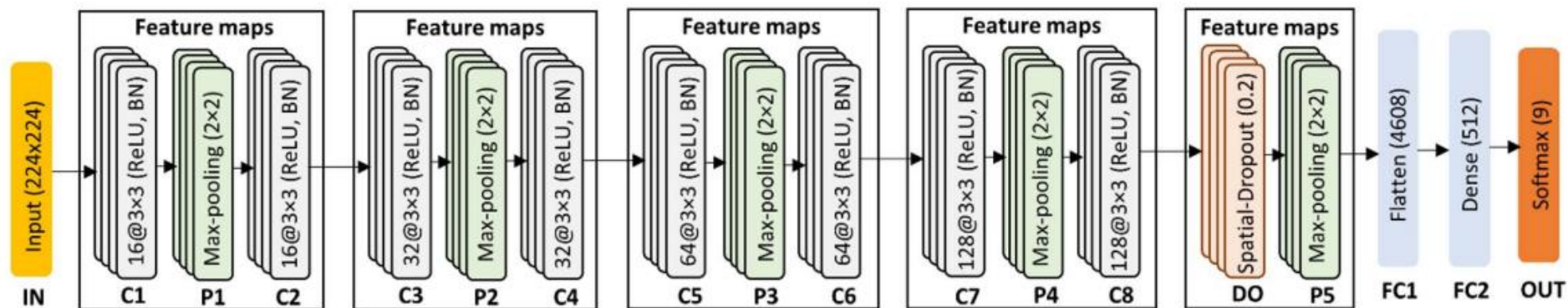
- 그래디언트(gradient) 값이 0에 가까워지는 것을 회피하기 위해
- 시그모이드(sigmoid) 함수 대신 **부분적 선형 함수**(linear function) 사용

### ▪ 부분적 가중치 학습

- 각 학습 단계에서 모든 가중치를 조정하는 대신
- 무작위로 선택한 **일부 가중치**만 조정

# 3. Proposed Deep CNN Model

## • 제안하는 CNN의 구조



\*Note: IN denotes input layer; C convolutional layer; P pooling layer; DO dropout layer; FC fully connected layer; OUT output layer; and BN batch normalization

- 하나의 입력층, 각각 배치 정규화(BN), 제로 패딩 및 ReLU(Rectified Linear Unit) 활성화가 있는 8개의 Conv 계층, 5개의 풀 계층 (4개의 Conv-Pool-Conv 그룹), 1개의 드롭아웃 계층, 2개의 완전연결(FC) 계층 및 1개의 출력층
- 첫 번째, 두 번째, 세 번째, 네 번째 Conv-Pool-Conv 그룹에 대해 각각 16, 32, 64, 128개의 피쳐맵을 사용
- VGP 해결
  - ReLu 활성화 함수 사용 (출력층만 softmax 함수)
  - Batch Normalization : 공분산의 변동을 줄임
- Overfitting 방지 : 규제화(regulation)
  - Dropout : 학습 시 일정 확률로 뉴런과 뉴런의 연결을 무작위로 제거시키는(drop-out) 것
  - SpatialDropout(SD) = 0.2 : SD는 Conv층에서  $n_f \times H \times W$  크기의 전체 피쳐맵을 삭제

# 3. Proposed Deep CNN Model

- 제안하는 CNN의 파라미터

TABLE I  
PROPOSED DEEP CNN MODEL PARAMETERS

Layer	Type	Feature Maps	Output Size	Filter size	Padding	Activation
IN	Input	3 (RGB)	224×224	-	-	-
C1	Convolution1	16	222×222	3×3	No	ReLU
P1	Max Pooling1	16	111×111	2×2	No	-
C2	Convolution2	16	111×111	3×3	Yes	ReLU
C3	Convolution3	32	111×111	3×3	Yes	ReLU
P2	Max Pooling2	32	55×55	2×2	No	-
C4	Convolution4	32	55×55	3×3	Yes	ReLU
C5	Convolution5	64	55×55	3×3	Yes	ReLU
P3	Max Pooling3	64	27×27	2×2	No	-
C6	Convolution6	64	27×27	3×3	Yes	ReLU
C7	Convolution7	128	27×27	3×3	Yes	ReLU
P4	Max Pooling4	128	13×13	2×2	No	-
C8	Convolution8	128	13×13	3×3	Yes	ReLU
P5	Max Pooling5	128	6×6	2×2	No	-
FC1	Fully-Connected1	1	4608	-	-	ReLU
FC2	Fully Connected2	1	512	-	-	ReLU
OUT	Output	1	9	-	-	Softmax

# 4. 실험 및 토의

## A. CNN-WDI 모델의 학습

### • 학습

- optimizer = Adam Stochastic
- batch size = 100
- epoch = 20
- learning rate = 0.001
- loss function = categorical crossentropy

### • 규제화(regulation)

- Batch Normalization
- Spatial Dropout = 0.2

- Results (결과) : 연구의 직접적인 결과(fact)
- Discussion (토의) : 저자의 논리적 설명/해석
- Conclusion (결론) : + 저자의 주장까지 제시

### • 컴퓨터 사양 (H/W)

- CPU : Intel Xeon CPU E5-2696 v5 @ 4.40GHz
- RAM : 512GB
- GPU : NVIDIA GeForce GTX 1080 24GB

### • AI Framework (S/W)

- Tensorflow 및 Keras

## 4. 실험 및 토의

### B. 성능 평가

- **Confusion matrix** (혼동/오차 행렬) : 분류기의 성능을 측정하는 도구

Predicted Class

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

정밀도

정확도

Recall 재현율

Trade-off

정밀도와 재현율의 조화평균 :  $F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$

## 4. 실험 및 토의

### c. 결과 및 분석

- 불균형 데이터셋과 균형잡힌 데이터셋에 대한 모델의 성능 비교
  - 불균형한 데이터셋이 있는 경우 대부분의 소수 클래스에 대해 매우 저조한 성능
- **결론1 : 데이터 증량방법이 효과적임을 증명**
  - CNN 모델은 대규모 데이터셋이 있을 때 더 나은 학습을 할 수 있기 때문

TABLE II  
PERFORMANCE EVALUATION OF CNN-WDI FOR BALANCED AND IMBALANCED DATASETS (%)

Defect Class	Balanced dataset			Imbalanced dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Center	93.6	<b>98.0</b>	95.7	<b>94.7</b>	97.7	<b>96.2</b>
Donut	<b>96.0</b>	<b>98.3</b>	<b>97.2</b>	85.2	90.4	87.7
Edge-Loc	<b>97.4</b>	<b>93.1</b>	<b>95.2</b>	89.2	87.4	88.3
Edge-Ring	94.4	91.7	93.0	<b>97.7</b>	<b>98.3</b>	<b>98.0</b>
Local	<b>91.6</b>	<b>90.3</b>	<b>90.9</b>	82.8	81.4	82.1
Near-Full	98.9	<b>99.9</b>	<b>99.4</b>	<b>100.0</b>	59.1	74.2
Random	<b>96.7</b>	<b>96.5</b>	<b>96.6</b>	83.0	90.0	86.4
Scratch	<b>98.1</b>	<b>98.7</b>	<b>98.4</b>	80.4	73.2	76.6
None	99.5	99.7	99.6	<b>99.9</b>	<b>100.0</b>	<b>100.0</b>
Average	<b>96.24</b>	<b>96.24</b>	<b>96.22</b>	90.32	86.39	87.72

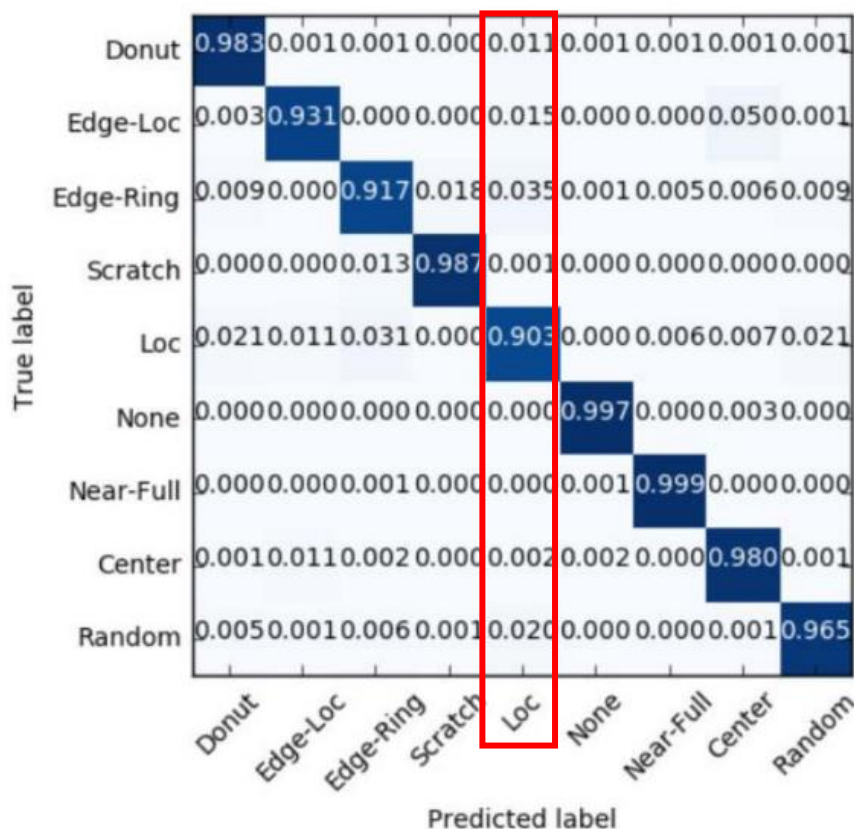
Note: Boldface numbers denote the highest values of different performance measures between balanced and imbalanced datasets

## 4. 실험 및 토의

### c. 결과 및 분석

- **Confusion matrix 결과로 구체적인 분석**

- 주대각선 값은 올바르게 분류된 비율
- Local 불량은 Donut, Edge-Loc 및 Random 불량 패턴과 매우 유사하므로 오분류가 많다.
- Edge-Loc 클래스는 각각 1.5% 및 5.0%의 비율로 Local 및 Center 클래스로 잘못 분류
- Edge-Ring 클래스는 Scratch 및 Local 클래스로 각각 1.8% 및 3.5%의 비율로 오분류





## 4. 실험 및 토의

### c. 결과 및 분석

- **규제화 기법 효과 입증**

- CNN-D (단순 Dropout) / CNN-BN / CNN-SD vs. CNN-WDI : BN + SD **(BN+D)?**

- **다른 구조의 모델과 비교하여 우수성 입증 ([저자의 이전 논문](#))**

- VGG-16 : 13ro의 Conv층, 5개의 max-pooling층, 3개의 완전연결층 (16층)

- ANN : 입력층-은닉층(100 노드)-출력층

- SVM : 수동으로 Radon 기반 특징 추출, OvR(one-vs-rest) 방법으로 다중 분류

- 학습 매개변수가 270만 개에 불과한 제안된 모델에 비해 VGG-16은 학습 매개변수의 수(즉, 1억 3,420만 개)가 매우 많지만, 여전히 우리 모델은 최대 18.6%의 학습 정확도를 개선 (학습시간도 1/3로 단축)

- **모든 성능지표에 대해 제안한 모델의 성능이 우수함**

TABLE III  
OVERALL PERFORMANCE COMPARISON OF VARIOUS CLASSIFIERS (%)

Classifier	Training Acc	Validation Acc	Testing Acc	Precision	Recall	F1-Score
CNN-WDI	98.9	<b>96.4</b>	<b>96.2</b>	<b>96.2</b>	<b>96.2</b>	<b>96.2</b>
CNN-D	97.6	95.5	95.2	95.2	95.2	95.2
CNN-BN	<b>99.4</b>	95.6	95.6	95.6	95.6	95.6
CNN-SD	98.6	94.7	94.8	94.8	94.8	94.8
VGG-16	82.3	80.0	80.1	80.3	80.1	79.9
ANN	95.9	95.9	72.0	95.2	95.9	95.4
SVM	91.3	91.0	32.6	87.5	91.0	88.0

Note: Boldface numbers denote the highest values of different performance measures and Acc accuracy



## 4. 실험 및 토의

### c. 결과 및 분석

- **이전의 SOTA(state-of-the-art) 모델과의 비교하여 우수성 입증**
  - 다른 모든 분류 모델은 일부 불량 클래스를 분류하는 정확도가 낮음
  - **WMFPR**은 Local 클래스에 대해 68.5%의 정확도
  - **WMDPI**는 Local 및 Scratch 클래스에 대해 각각 60.0% 및 34.0%의 정확도
  - **DTE-WMFRP**는 Edge-Loc 및 Local 클래스에 대해 모두 83.5%의 정확도를 보임
- **이전의 제일 높은 평균 정확도를 6.4%까지 향상 → 모델의 우수성 입증**

TABLE IV  
COMPARISON OF DEFECT CLASSIFICATION RESULTS FOR DIFFERENT CLASSIFICATION MODELS (%)

Classification model	Center	Donut	Edge-Loc	Edge-Ring	Local	Random	Scratch	Near-Full	None	Average
CNN-WDI	<b>98.0</b>	<b>98.3</b>	<b>93.1</b>	91.7	<b>90.3</b>	<b>96.5</b>	<b>98.7</b>	<b>99.9</b>	99.7	<b>96.2</b>
WMFPR [3]	84.9	74.0	85.1	79.7	68.5	79.8	82.4	97.9	95.7	83.1
WMDPI [6]	86.0	77.0	77.0	<b>95.0</b>	60.0	94.0	34.0	97.0	<b>100.0</b>	80.0
DTE-WMFPR [5]	95.8	92.2	83.5	86.8	83.5	95.8	86.0	N/A	100.0	90.4

Note: Boldface numbers denote the highest values of testing accuracies for various classification models

# 4. 실험 및 토의

## c. 결과 및 분석

- epoch에 따른 학습/검증 정확도 및 손실(loss) 추이 (제안모델 vs. VGG-16)

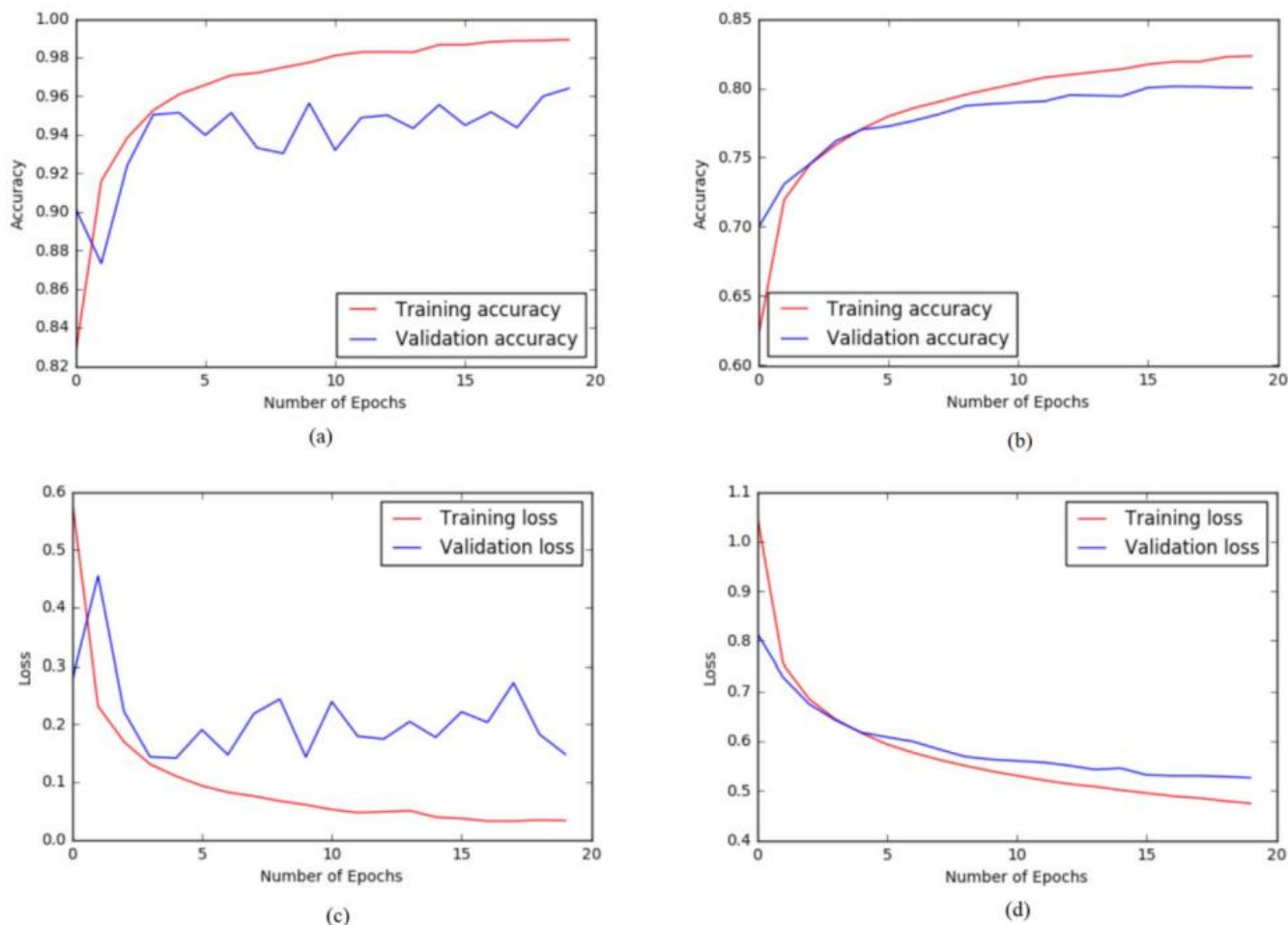


Fig. 5. The performance results of CNN-WDI model and VGG-16 model: (a) The accuracies of CNN-WDI, (b) The accuracies of VGG-16, (c) The losses of CNN-WDI, and (d) The losses of VGG-16.

## 5. 결론

반도체 엔지니어는 전문 지식이나 경험적 지식 없이 웨이퍼 불량률의 조기 진단을 위해 자동 웨이퍼 분류 기술을 사용합니다. 기존의 웨이퍼 불량 분석은 대부분 수동 특징 추출과 많은 하이퍼파라미터 설정이 필요한 머신러닝 기반 분류 모델을 적용한 반면 CNN 모델은 다양한 불량 클래스의 유효 특징을 자동으로 추출할 수 있습니다. 본 논문에서는 반도체 제조공정에서 웨이퍼 맵 불량을 분류하기 위해 딥러닝 기반 CNN-WDI 모델을 제안하였습니다. 우리는 9개의 다른 불량 패턴을 포함하는 WM-811K의 실제 웨이퍼 맵 데이터셋을 사용했습니다. 이 데이터셋은 불균형이 심하므로 이 문제를 해결하기 위해 데이터 증량 기법을 적용했습니다. 일련의 컨볼루션 및 풀링 계층을 적용하여 원시 웨이퍼 이미지에서 중요한 특징을 추출했습니다. 제안한 모델의 규제화를 위해 배치 정규화와 공간적 드롭아웃 방법을 사용하여 모델의 학습 속도와 분류 정확도를 향상시켰습니다. 우리 모델은 동일한 데이터셋을 사용하여 기존에 제안된 WMFPR, DTE-WMFPR, WMDPI 등의 모델에 비해 모든 불량 클래스에 대해 매우 높은 성능을 보였고 평균 96.2%의 분류 정확도를 달성했습니다. CNN-WDI는 VGG-16, SVM 및 ANN 분류기를 능가했습니다. 향후 연구를 위해 동일한 웨이퍼 이미지에서 다중 불량을 추출하여 분류 정확도를 개선할 것입니다.

1. 배경(서론)
  - 정의, 목적
  - 중요성
  - 문제점
2. 제안(해법)
  - CNN
  - 증량
  - 자동 특징추출
  - 배치정규화
  - 드롭아웃
3. 결과
  - 우수성
  - 기여
4. 향후 연구

**Q & A**