# wafer map defect classification by deep learning

조원 : 김영훈, 고영국, 문석진
팀명 : 원미 엔지니어링
지도교수 : 김태선교수님

# 목차

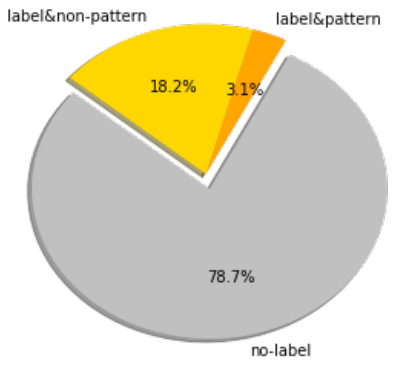
# 프로젝트 개요

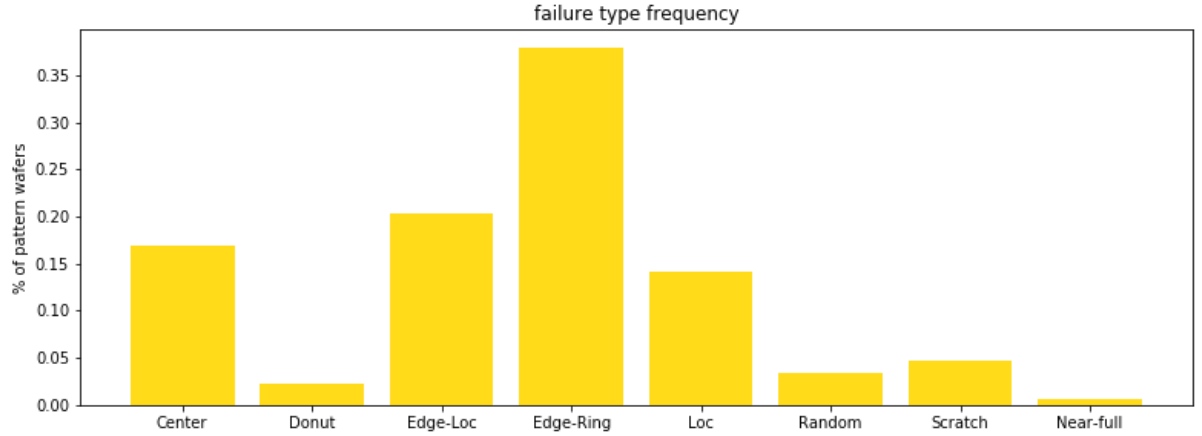
- 학습되지 않은 웨이퍼 맵 표면의 밀도기반 불량을 딥러닝을 이용해 빠르게 분류한다.



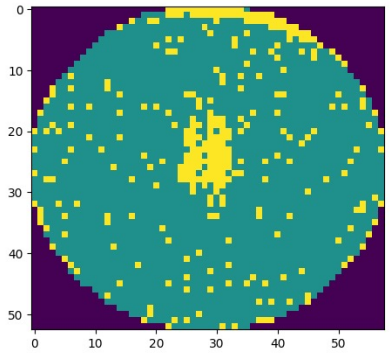
# 데이터셋 분석

- WM-811K
  - real data from TSMC
  - 811,457 wafer maps from 46,293 lots
  - only about 20% were labeled (172,951 / 811,457)
- 9 types (8 defect types + 1 non defect type)

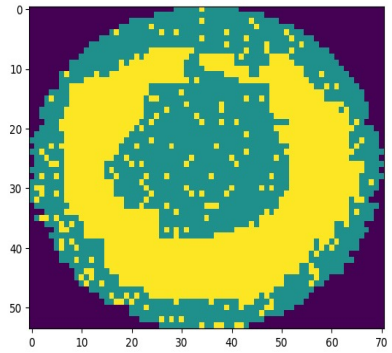


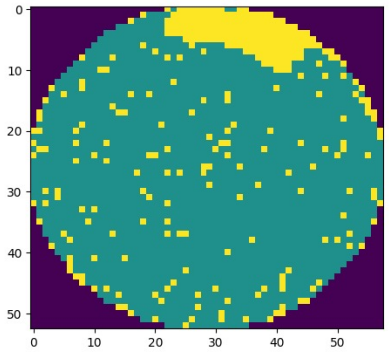
# 데이터셋 분석 (레이블링)



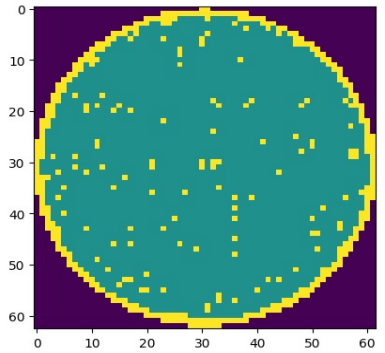
center



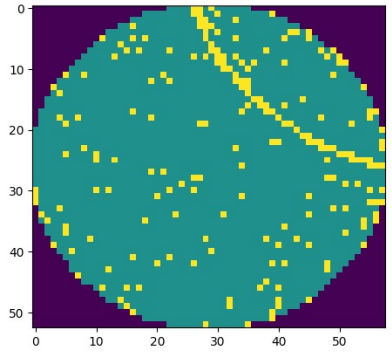
donut



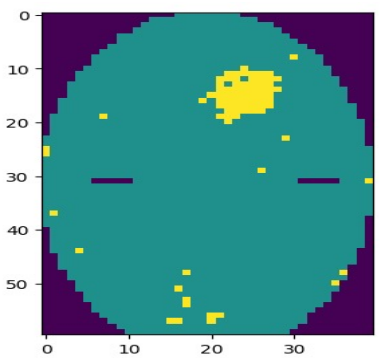
Edge-local



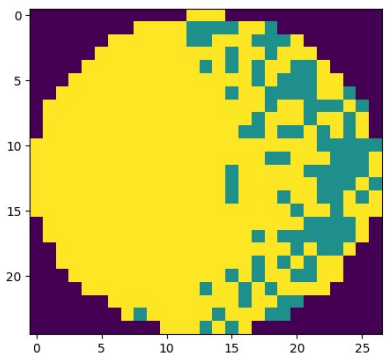
Edge-ring



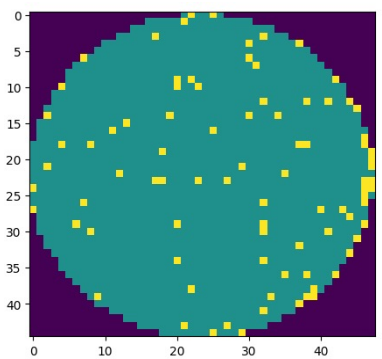
scratch



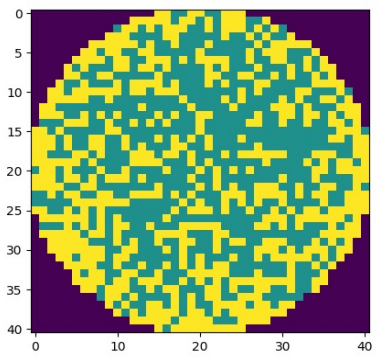
local



Near-full



none



random

# 데이터셋 분석 (정성적 레이블링의 근거)[1]

- center : 불량 패턴이 웨이퍼 정 가운데에 위치한 경우
- edge : 불량 패턴이 웨이퍼 가장자리에 위치한 경우
- scratch : 불량 패턴이 얇은 선의 형태로 위치한 경우
- edge-ring : 불량 패턴이 안이 고리 형태 혹은 고리의 일부의 형태를 가질 경우
- local : 불량 패턴이 볼록하며(Convex) 두께 있는 덩어리 형태를 가질 경우
- edge-local : 불량 패턴이 웨이퍼 가장자리에 위치하며, 볼록하며(Convex) 두께 있는 덩어리 형태를 가질 경우
- none : 웨이퍼 맵 상에 군집을 이룬 불량 칩이 없을 경우
- near-full : 웨이퍼 맵 상에 군집을 이룬 불량 칩이 대부분 일 경우
- random : 웨이퍼 맵 상에 군집을 이룬 불량 칩이 near-full보다 적고 none보다 많을 경우

- [1] '웨이퍼 맵의 정량적 분류 체계 개발’ ,포항공과대학교 산업경영공학과 ,최승현 외

# 데이터셋 분석 (결함원인과 레이블링 연결)[2]

- center type : due to abnormality of RF(Radio Frequency) power, abnormality in liquid pressure, due to velocity on the outside of the wafer is fast.[3]
- local type : due to slit valve leak, abnormality during robot handsoff or abnormality in the pump.
- random type : due to contaminated pipes, abnormality in showerhead, or abnormality in control wafers.
- scratch type : mainly due to abnormality during robot handsoff or wafer impacts.

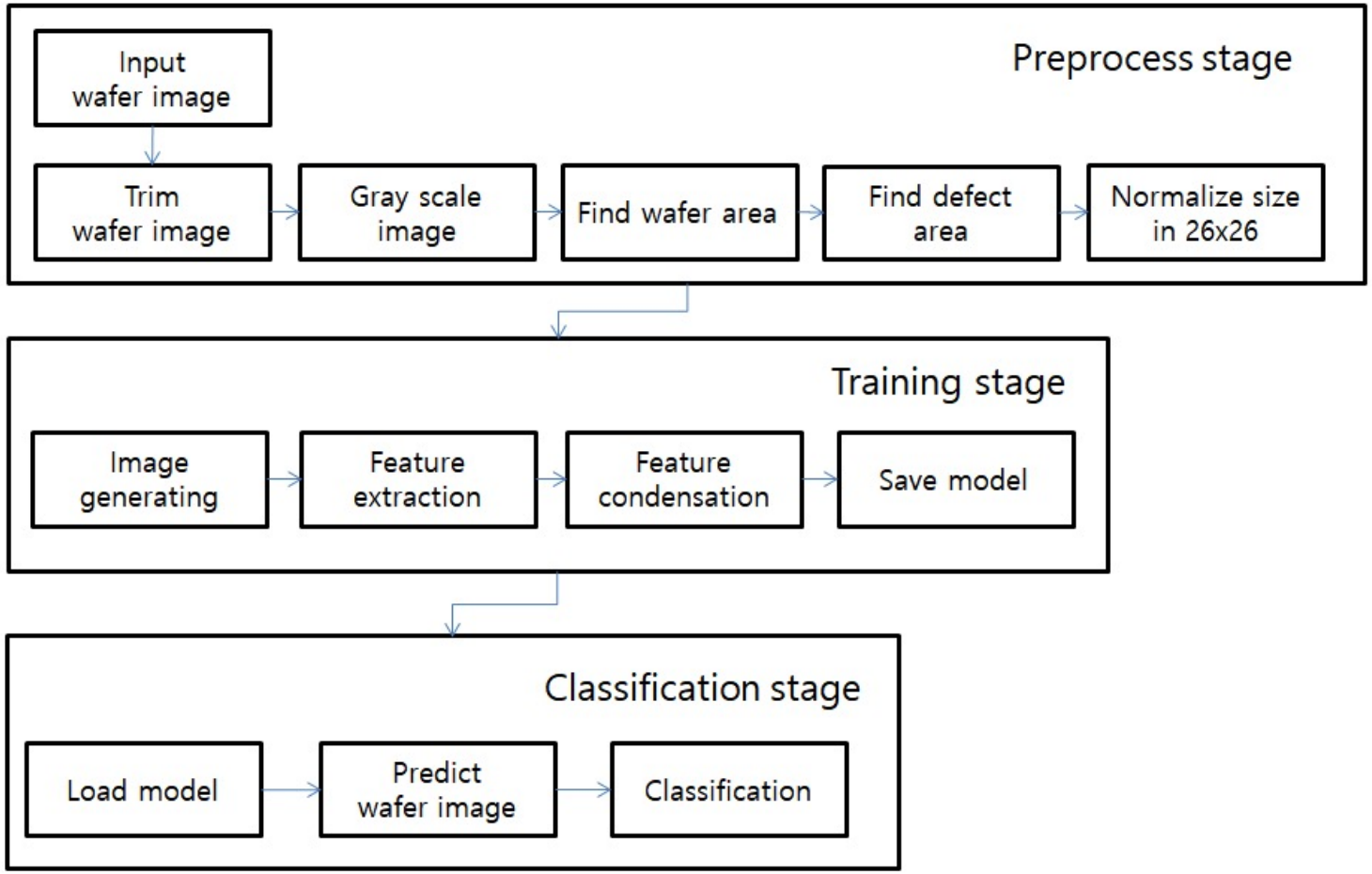
[2] 'Inspection and Classification of Semiconductor wafer Surface Defects Using CNN Deep Learning Networks, 2020, Jong-chih chein at el.
[3] 'bull's eye effects를 줄이기 위한 CMP System의 최적화 설계에 관한 연구', Byung Hoon Jeong at el.

# 데이터셋 분석 (결함원인과 레이블링 연결)

- donut type : Plasma-induced oxide damage can be modelled as damage produced by electrical current or voltage stress. Plasma processing causes MOSFET parameter degradation, from which one can deduce the plasma charging current. The effect of plasma etching on silicon–oxide interface reliability is also presented. The interface traps generated by plasma processing can be passivated with a forming gas anneal.
- edge-ring type : due to polishing pressure, the velocity of the polishing head and the table, the processing time for the wafer, the chemical reactions occurring on the wafer surface, the hydrodynamic condition of the slurry, and the properties of the pad.[5]

[4] ‘Thin gate oxide damage due to plasma processing’ , H C Shin and Chenming Hu 1996 *Semicond. Sci. Technol.* **11** 463
[5] ‘Planarization of Wafer Edge Profile in Chemical Mechanical Polishing’, Yeongbong Park at el.

# 전체 flow에 대한 요약



# 데이터셋 전처리

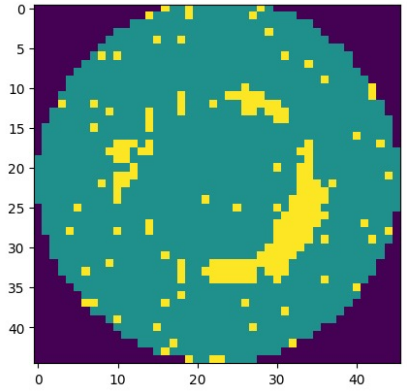
| | |
|---|---|
| Center : 90 | Center : 2160 |
| Donut : 1 | Donut : 2002 |
| Edge-Loc : 296 | Edge-Loc : 2368 |
| Edge-Ring : 31 | Edge-Ring : 2046 |
| Loc : 297 | Loc : 2376 |
| Near-full : 16 | Near-full : 2032 |
| Random : 74 | Random : 2146 |
| Scratch : 72 | Scratch : 2088 |
| none : 84 | none : 2100 |

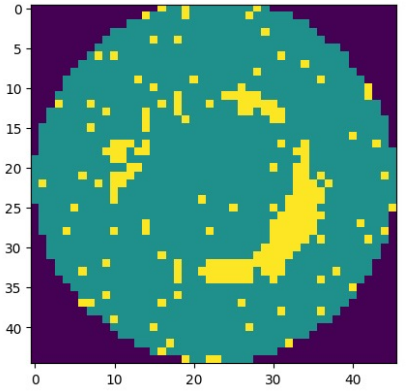
- 데이터셋 전처리의 필요성
  - 데이터셋은 TSMC의 WM-811K를 이용한다.
    클래스 불균형이 있는 경우 개수가 적은 레이블에 대해 정확도가 낮아진다.
- 데이터셋 전처리 구현 방법

  - input wafer encode => dummy array 제작 => noised wafer를 2000개 생성하고 decode => 같은 길이로 label wafer 생성 => dummy data에 input.

➢ 압축 후 압축을 풀 때 자동으로 생성되는 noise를 이용한 방식입니다.

일반적인 keras image generator 함수(회전, 확대, crop 방식) 대신 이용했는데, 이 방식이 더욱 정확도 향상에 도움을 줍니다.
=> 이유 : wafer의 원형 형식은유지.



# 데이터셋 분리

(code)

```
x_train, x_test, y_train, y_test = train_test_split(new_X, new_Y,
                                                    test_size=0.33,
                                                    random_state=2019)
```

- Python , train_test_split 함수를 이용하여 train data와 test data를 2:1로 분리

# 모델,하이퍼 파라미터

- overfitting을 예방하는 모델 사용

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 256)               295168
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 2313
=================================================================
Total params: 307,625
Trainable params: 307,625
Non-trainable params: 0
_________________________________________________________________
```

conv2d : 필터를 이용하여 영상 특징을 추출합니다.

maxpooling2d : 영상에서의 사소한 변화가 특징 추출에 크게 영향을 미치지 않도록 합니다.

flatten : 2차원의 특징맵을 전결합층으로 전달하기 위해서 1차원 형식으로 바꿔줍니다.

dropout : 과적합을 방지하기 위해서 학습시에 지정된 임의의 비율만큼 임의의 입력 뉴런(2차원)을 제외시킵니다.

[6] https://tykimos.github.io/DeepBrick/

# k-fold cross validation을 통한 모델평가

- 3-fold cross validation 진행 (K=3)

Epoch 10/10

- 3s - loss: 0.0277 – acc: 0.9934

Epoch 10/10

- 3s – loss: 0.0132 – acc: 0.9972

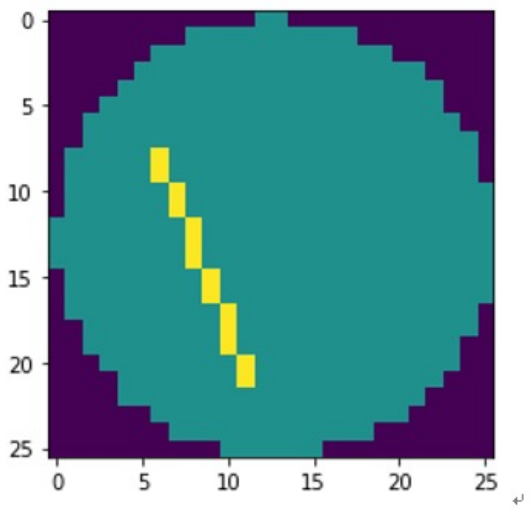
Epoch 10/10

- 3s – loss: 0.0422 - acc: 0.9823

Cross validation score : 0.9909

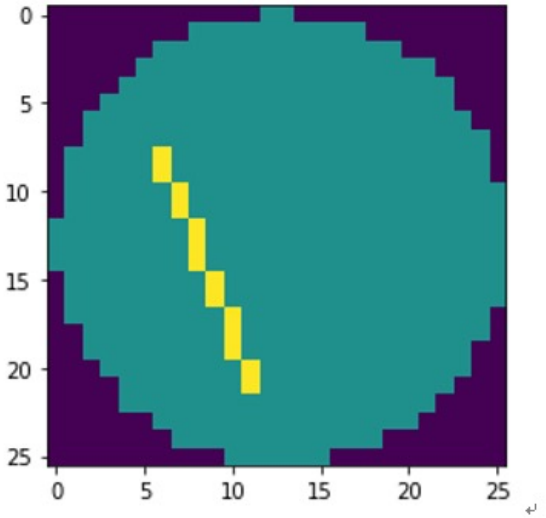
- 3-fold cross validation 결과 model이 적절하게 설정되었음을 확인가능.

# 모델,하이퍼 파라미터,Over fitting에 대한 문제 해결




```
[8.5853378e-04 6.8418416e-07 4.4919048e-03 1.6002790e-05 3.1817202e-02
 3.7514314e-10 3.6270215e-10 1.8576926e+00 9.8105118e+01]
Defect: None (98.11%)
```

< epoch=30 >




```
[4.6190261e-04 3.9668606e-09 2.2423232e-02 6.0229040e-06 2.9648530e-01
 5.6812040e-09 4.8111321e-08 5.4835911e+01 4.4844715e+01]
Defect: Scratch (54.84%)
```

< epoch=15 >

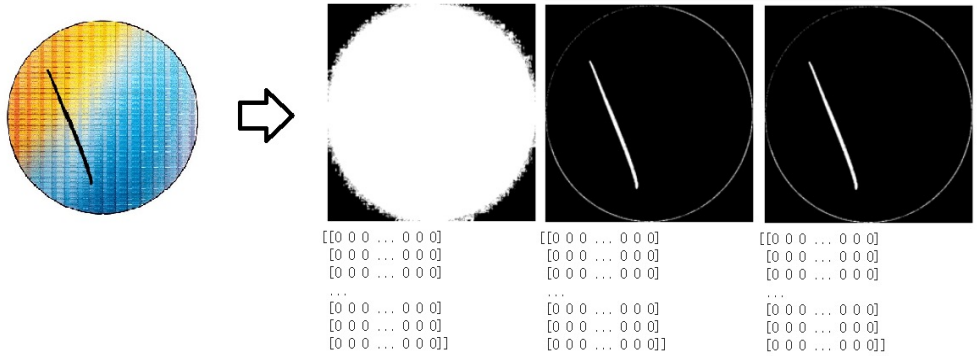
(상단의 wafer map은 untrained wafermap)

# 성능평가 (정확도) (수정)

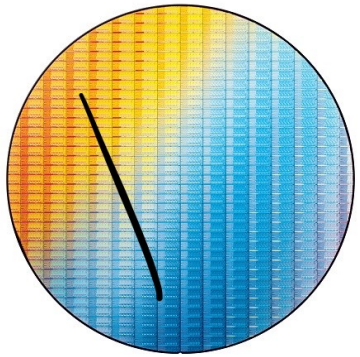
- ……
- Epoch 11/15
- 809/809 [==============================] - 3s 4ms/step - loss: 0.0436 - accuracy: 0.9842 - val_loss: 0.0274 - val_accuracy: 0.9929
- Epoch 12/15
- 809/809 [==============================] - 3s 4ms/step - loss: 0.0459 - accuracy: 0.9852 - val_loss: 0.0320 - val_accuracy: 0.9918
- Epoch 13/15
- 809/809 [==============================] - 3s 4ms/step - loss: 0.0400 - accuracy: 0.9860 - val_loss: 0.0236 - val_accuracy: 0.9937
- Epoch 14/15
- 809/809 [==============================] - 3s 4ms/step - loss: 0.0321 - accuracy: 0.9910 - val_loss: 0.0199 - val_accuracy: 0.9956
- Epoch 15/15
- 809/809 [==============================] - 3s 4ms/step - loss: 0.0291 - accuracy: 0.9898 - val_loss: 0.0235 - val_accuracy: 0.9933

- 매우 우수한 정확도인 0.9933 제공 , tensorflow의 check point 기능을 이용해 과적합을 최소화시킨 모델은 0.9956의 정확도 제공

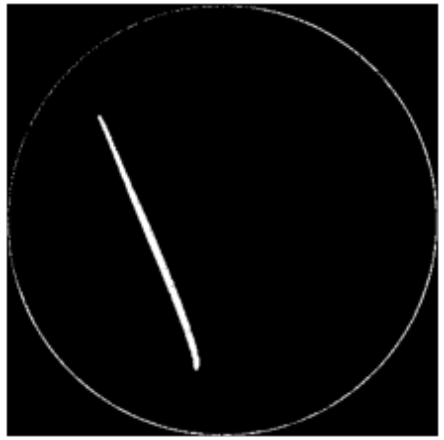
# 훈련되지 않은 이미지 전처리



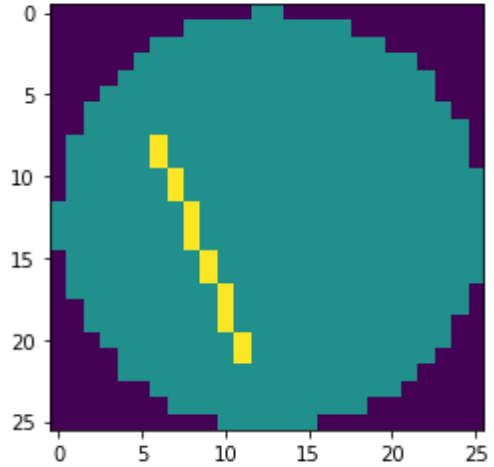
< 3차원배열화 >



< input image >



< input image processing >




< input image array 를 image화 >

# 정성적 레이블링 기준에 따라 만들어진 image에 대한 성능 평가 ,정확도

| 예측/입력 | center | donut | edgeloc | edgering | local | nearfull | random | scratch | none | |
|---|---|---|---|---|---|---|---|---|---|---|
| center | 9 | | | | 1 | | | | | |
| donut | | 0 | | | | | | | | |
| edgeloc | | | 8 | 1 | | | 1 | | | |
| edgering | | | | 9 | | | | | | |
| local | | 5 | 2 | | 9 | | | | | |
| nearfull | | | | | | 0 | | | | |
| random | | 2 | | | | 10 | 9 | | | |
| scratch | 1 | 3 | | | | | | 10 | | |
| none | | | | | | | | | 10 | |
| | | | | | | | | | | 74/90 |

- 정확도 : 82% ( vs 99.3% )
- donut , near-full 분류 실패이유

  도넛 패턴을 가진 웨이퍼맵이 1개였으므로 이미지개수를 증식해도 유사성을 띄는 이미지만이 증식되기 때문에 데이터셋이 충분히 확보되었다고 보기 어렵다.

# 분류 시연 동영상

```
import numpy as np
from keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/drive')

# 분류 가능하게 이미지 가공
    # img1 = image.load_img('/content/drive/My Drive/input/test4.jpg'.format(test_img), target_size=(64, 64))
    # img = image.img_to_array(img1)
    # img = img/255
    # img = np.expand_dims(img, axis=0)

# -*- coding: utf-8 -*-
"""
Created on Wed Jan 20 18:05:48 2021

@author: Owner
"""
import cv2, sys
import numpy as np
from matplotlib import pyplot as plt


imageFile='/content/drive/My Drive/input/input2/WaferMap/balanced/Center/center_110.jpg'
img = cv2.imread(imageFile, cv2.IMREAD_COLOR)



blur = cv2.GaussianBlur(img, ksize=(3,3), sigmaX=0)
ret, thresh1 = cv2.threshold(blur, 127, 255, cv2.THRESH_BINARY)
edged = cv2.Canny(blur, 10, 250)

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7,7))
closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
```

# 분류 실패 케이스 분석

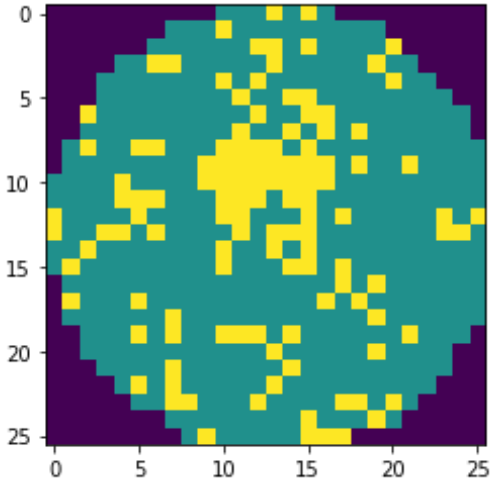
- 분류 실패 케이스 분석
  - 정성적 레이블링 기준에 따른 분류실패발생

[6.3547630e+01 8.1575481e-04 7.1986988e-02 3.1581279e-04 2.9763290e+01
8.4340124e-04 4.4737205e-02 6.5392623e+00 3.1117633e-02]

defect: center (63.55%) , local(29.76%)

정성적 레이블링을 진행했을 때 center의 특성과
local의 특성을 모두 띄고 있어 분류에 실패한 것으로 예상.

29.7%의 확률로 local로 분류하였다.



# (Donut, Near-full 제외)
# 정성적 레이블링 기준에 따른 image에 대한 성능평가

| 예측/입력 | center | edgeloc | edgering | local | random | scratch | none | |
|---|---|---|---|---|---|---|---|---|
| center | 9 | | | 1 | | | | |
| edgeloc | | 8 | 1 | | 1 | | | |
| edgering | | | 9 | | | | | |
| local | | 2 | | 9 | | | | |
| random | | | | | 9 | | | |
| scratch | 1 | | | | | 10 | | |
| none | | | | | | | 10 | |
| | | | | | | | | 64/70 |

- 정확도 : 91%
- donut , near-full 의 경우 데이터셋의 신뢰성이 확보되지 않았으며 성능 평과 결과 분류가 불가능한 것으로 판단되었음.
- donut , near-full 데이터셋이 충분히 확보되어 신뢰성이 보장된다면 분류에 성공할 것으로 예상됨.

# 결론

- 훈련데이터를 train set과 test set으로 나누고 훈련하여 측정한 정확도는 훈련되지않은 데이터에 대한 분류 정확도와 비례하지 않는 것을 확인하였다. 그 이유를 overfitting과 데이터셋의 부족으로 예측하였다.
- overfitting을 최소화 시키는 모델 설계 , 충분한 데이터셋 확보 , 정량적 레이블링 , 레이블링 세분화를 진행한다면 훈련되지 않은 이미지에 대한 정확도가 훈련데이터간 정확도에 근접하게 향상될 것으로 예상
- 기존의 정확도만을 높이는 모델 사용은 딥러닝을 이용한 결함분류시스템에 사용하기 적합하지 않음.
- 웨이퍼 맵 분류 체계 도출 시 정량적 기준으로 분류 기준 정의를 제안한다.
  ex) Scratch : 웨이퍼 맵이 얇은 선의 형태로 위치한 경우

  ->5개 이상의 불량 칩이 일렬로 붙어 있을 경우 Scratch로 분류
  장축을 기준으로 최대 두께가 4개 불량 칩 이상일 경우 local
  으로 분류[1]
- 충분한 데이터셋을 보유하기 어려운 경우 정량적 기준에 따른 데이터 생성과 포아송 분포를 사용한 랜덤결함패턴을 결합한 방법을 제안한다.[7]

[7] ‘Wafer Map Defect Pattern Classification and Image Retrieval Using Convolutional Neural Network’ , Takeshi Nakazawa and Deepak V. Kulkarni

# 프로젝트의 기대효과

- 밀도기반 불량 발생을 조기에 인지하고 분류함으로 전체적인 수율 향상에 도움을 준다.

- 검사과정 자동화를 통한 생산성, 경제성 증대

- 육안으로 판단하기 애매한 불량에 대해 경향성을 제공함으로서, 단위공정 개선의 우선순위를 부여한다.
  Ex) 정성적 레이블링 기준에 따른 분류실패발생