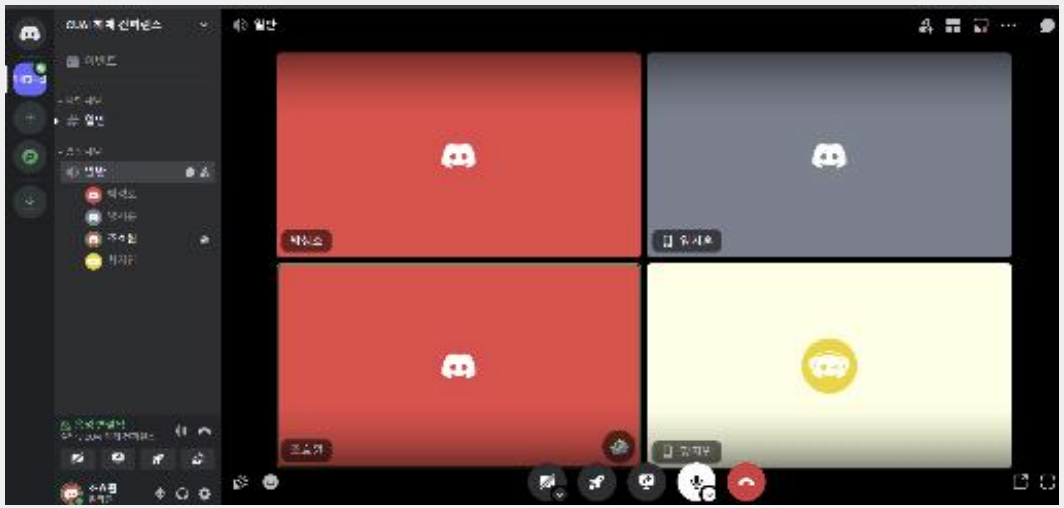
CUAI 스터디 하계 DA 1팀

2024.07.30

발표자 : 황지민

스터디원 소개 및 만남 인증



스터디원 1 : 박성호

스터디원 2 : 박준상

스터디원 3 : 양지훈

스터디원 4: 조효원

스터디원 5: 황지민

데이터 분석 주제 변경

- **WM-811K wafer map**

Data Card | Code (19) | Discussion (4) | Suggestions (0)

About Dataset

Information paper:

- W., Ming-Ju, Jyh-Shing R. Jang, and Jui-Long Chen. "Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets." IEEE Transactions on Semiconductor Manufacturing 28, no. 1 (February 2015): 1–12.

Usability

5.00

License

CC0: Public Domain

Expected update frequency

Not specified

Tags

- **변경 이유**

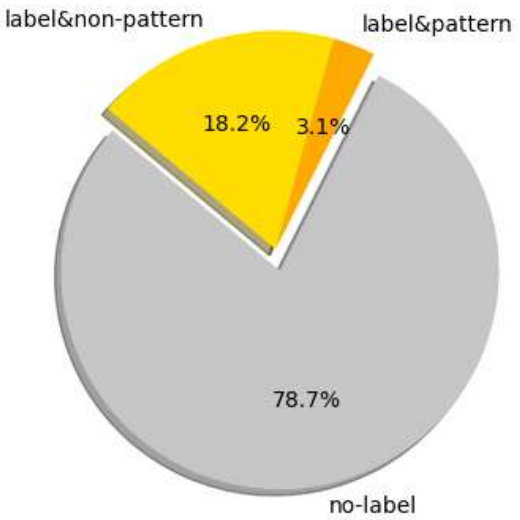
기존 데이터의 신뢰성 문제

스마트 팩토리의 다양한 센서에서 취득한 공정 데이터

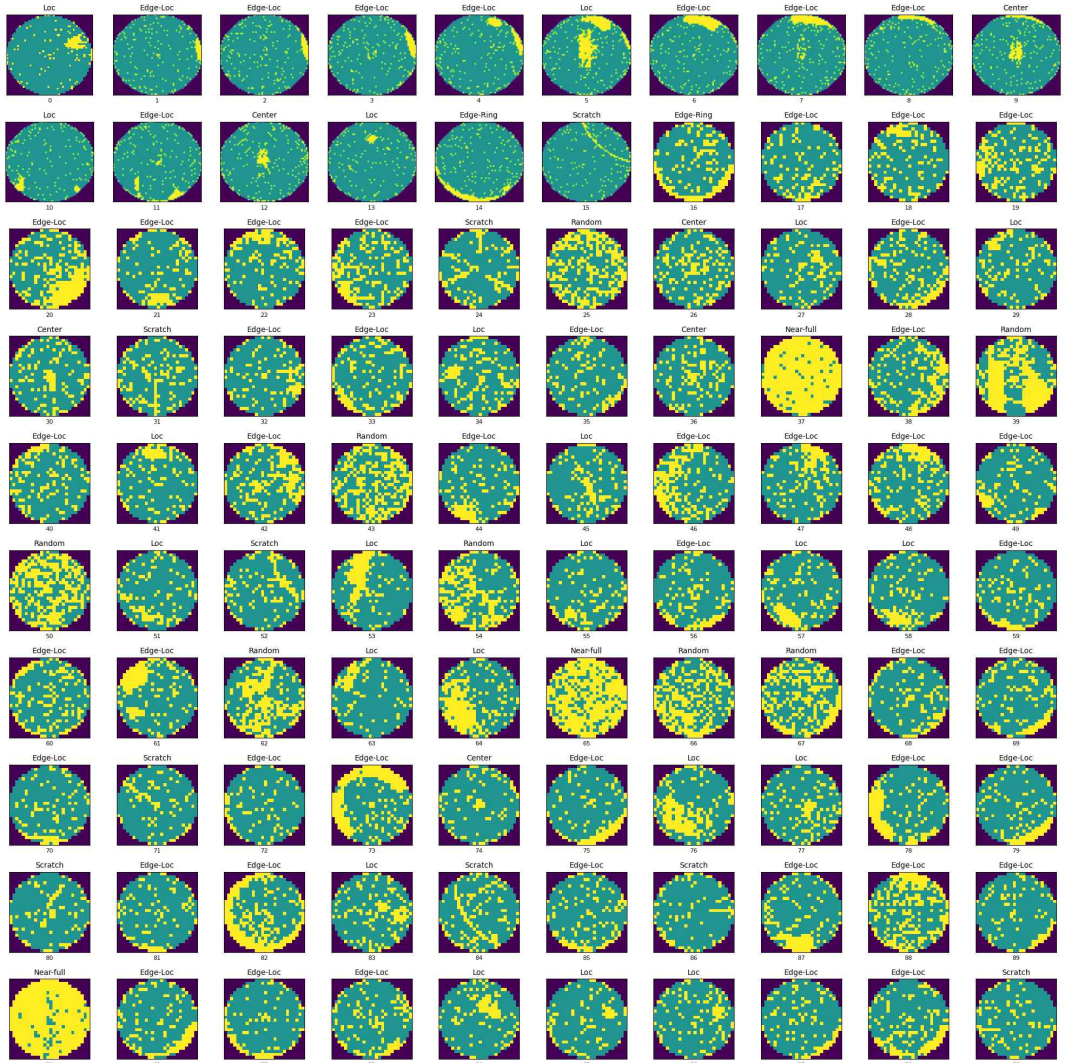
데이터 설명

0: None
1: 결함이 없는 die
2: 결함이 있는 die

	waferMap	dieSize	lotName	trianTestLabel	failureType	waferMapDim
526059	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	515.0	lot32900	[]	[]	(25, 27)
450122	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2,...	712.0	lot27611	[]	[]	(32, 29)
407908	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	7870.0	lot24369	[]	[]	(63, 160)
533958	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	710.0	lot33375	[]	[]	(32, 29)
316026	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1226.0	lot19233	[]	[]	(40, 40)

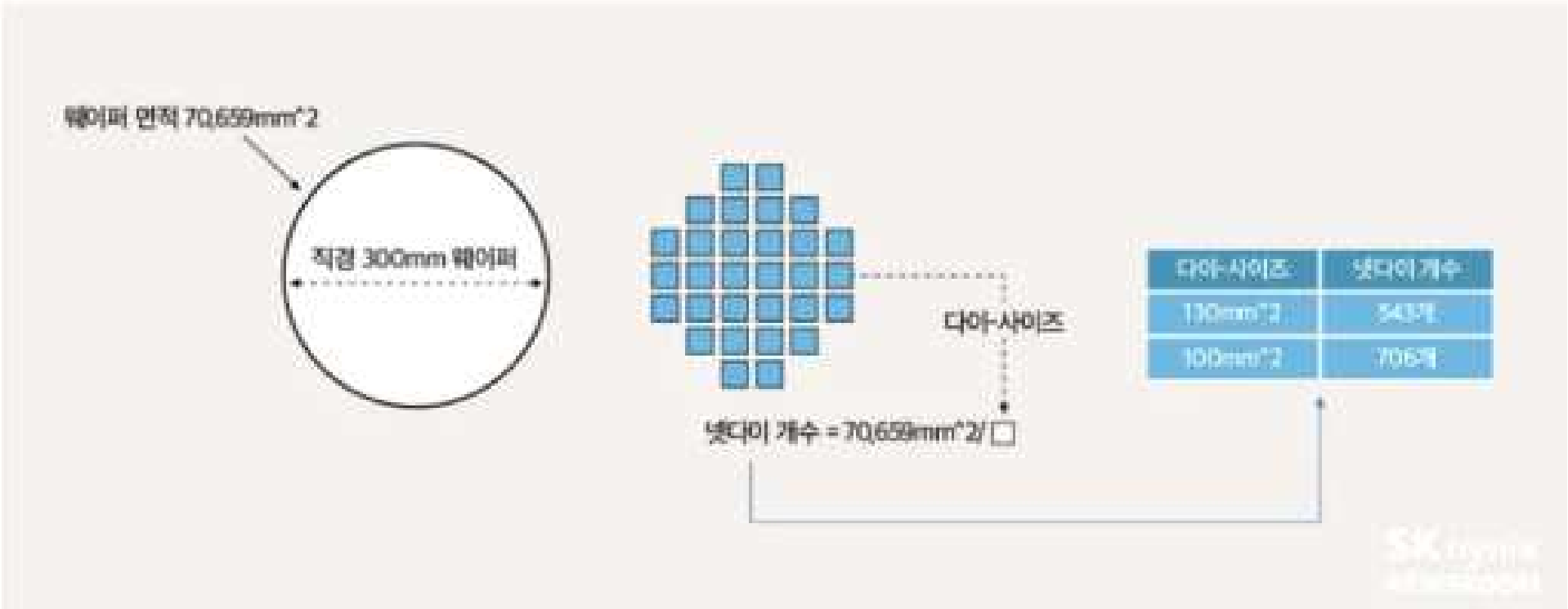


이미지 데이터 시각화



웨이퍼? 다이?

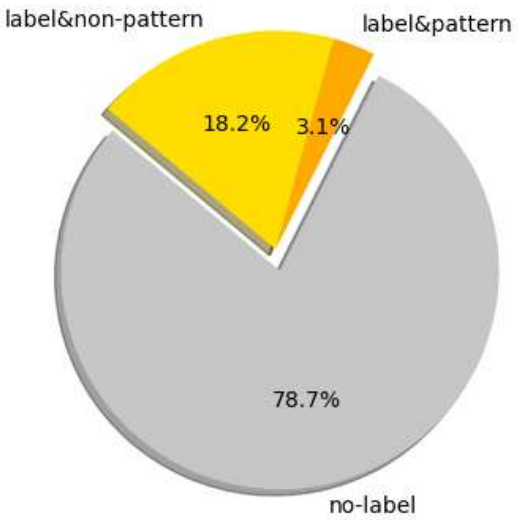
2. 넷다이(Net Die)와 다이-사이즈(Die-Size)



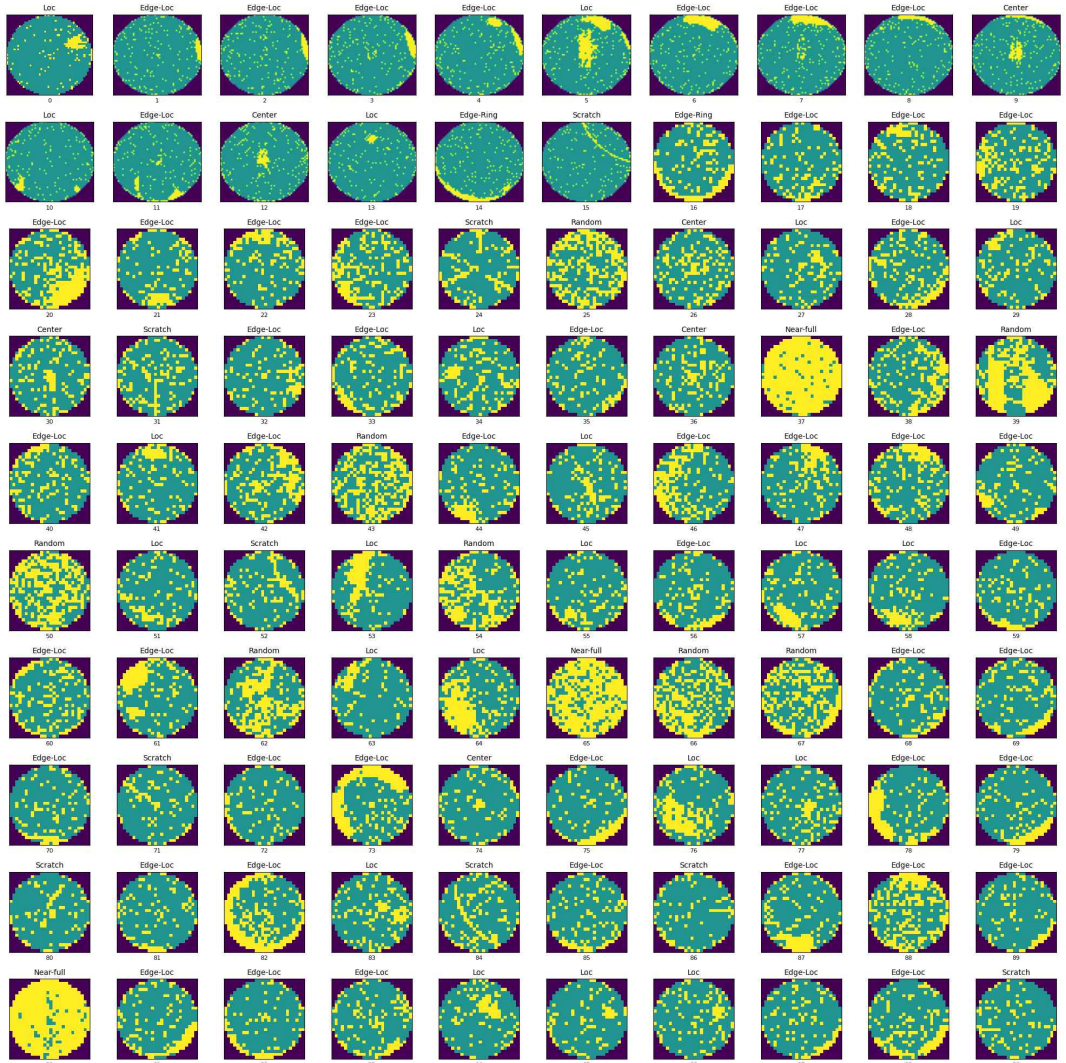
데이터 설명

0: None
1: 결함이 없는 die
2: 결함이 있는 die

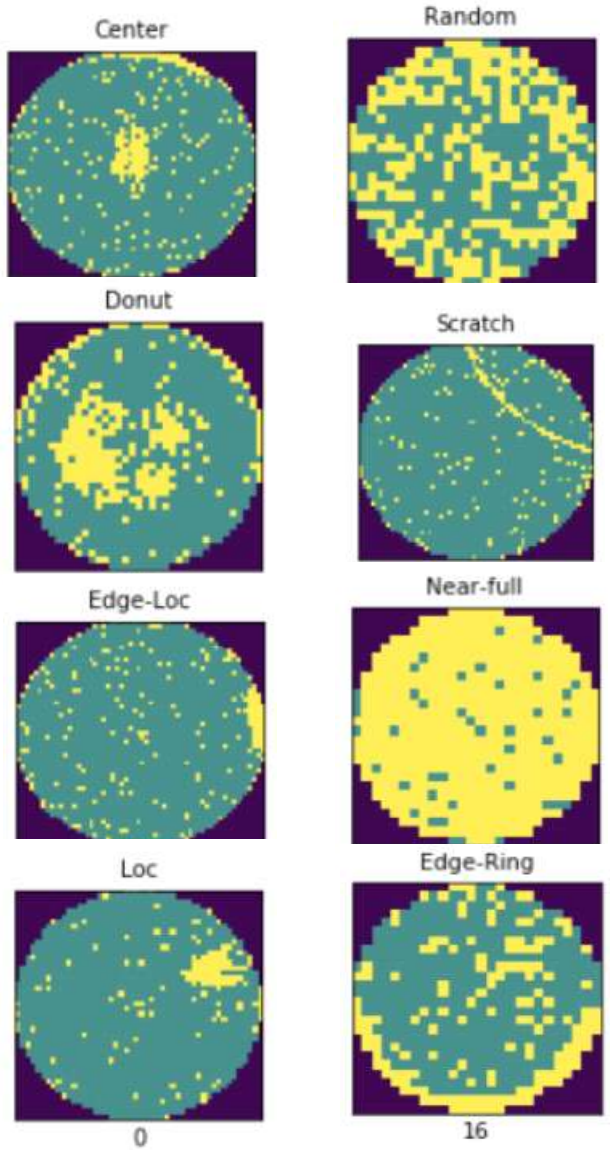
	waferMap	dieSize	lotName	trianTestLabel	failureType	waferMapDim
526059	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	515.0	lot32900	[]	[]	(25, 27)
450122	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2,...	712.0	lot27611	[]	[]	(32, 29)
407908	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	7870.0	lot24369	[]	[]	(63, 160)
533958	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	710.0	lot33375	[]	[]	(32, 29)
316026	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1226.0	lot19233	[]	[]	(40, 40)

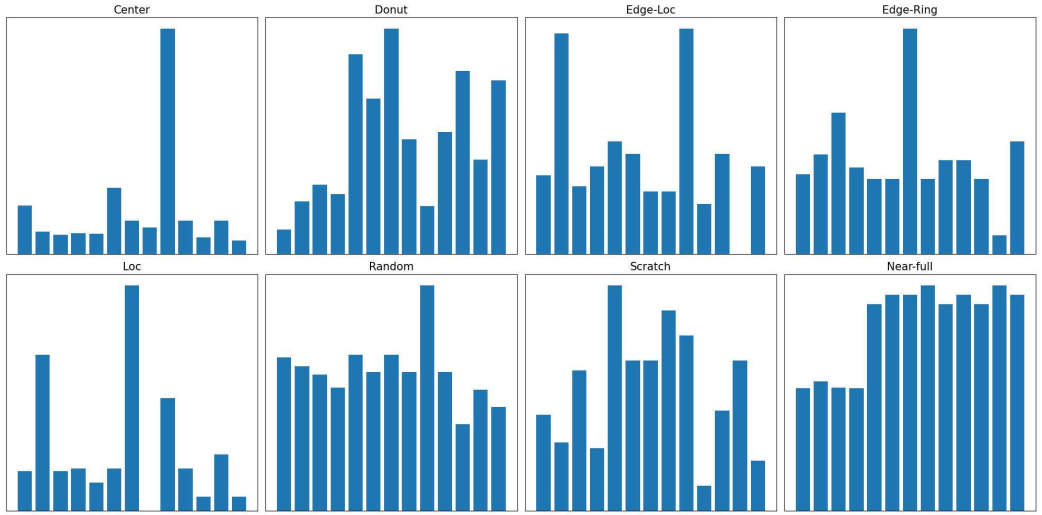


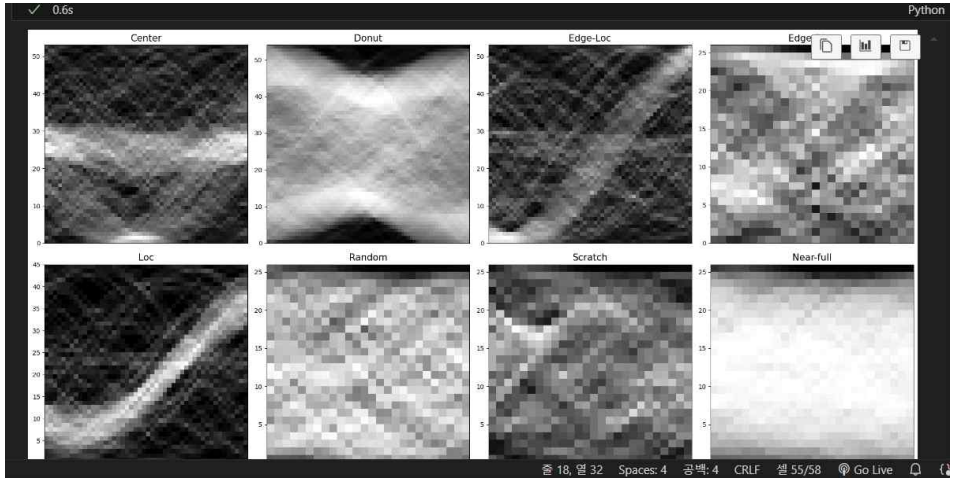
이미지 데이터 시각화



데이터 설명


Center
Random
Donut
Scratch
Edge-Loc
Near-full
Loc
Edge-Ring
0
16





선형 커널 (Linear Kernel)	$x,y) = (x \cdot y)$
다항식 커널 (Polynomial Kernel)	$K(x,y) = (x \cdot y+1)$
가우시안 커널 (Radial Basis Function(Gaussian))	$K(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma}}$
시그모이드 커널 (Sigmoid Kernel)	$K(x,y) = \tanh(\theta_1 x \cdot y+\theta_2)$

Table 3.1 커널법을 활용한 비선형 변형

```
# ---multiclass classification ---#
# One Vs One
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
clf2 = OneVsOneClassifier(LinearSVC(random_state = RANDOM_STATE)).fit(X_train, y_train)
y_train_pred = clf2.predict(X_train)
y_test_pred = clf2.predict(X_test)
train_acc2 = np.sum(y_train == y_train_pred, axis=0, dtype='float') / X_train.shape[0]
test_acc2 = np.sum(y_test == y_test_pred, axis=0, dtype='float') / X_test.shape[0]
print('One-Vs-One training acc: {}'.format(train_acc2*100)) #One-Vs-One training acc: 80.45
print('One Vs One Testing acc: {}'.format(test_acc2*100)) #One Vs One Testing acc: 79.04
print("y_train_pred[:100]: ", y_train_pred[:100])
print ("y train[:100]: ", y_train[:100])
```

5.2s

```
One Vs One Training acc: 82.74204503892575
One-Vs-One Testing acc: 82.30244514406583
y train pred[:100]:  [4 0 2 2 2 3 2 0 2 2 6 0 4 0 1 4 3 2 2 5 2 2 0 1 3 0 3 3 3 1 5 3 4 1 4 3 3
 4 3 2 2 2 3 3 0 6 3 5 2 3 0 3 2 2 2 0 3 3 1 0 4 2 3 3 3 3 4 3 0 3 2 4 0 2
 4 1 1 0 3 0 2 4 1 1 2 1 0 0 4 0 0 1 0 1 0 1 1 2 4 2]
y_train[:100]:  [5 0 4 2 0 3 2 0 2 2 6 0 4 0 3 0 6 2 2 5 2 4 0 5 3 0 3 3 3 3 2 3 4 3 4 3 3
 4 3 2 2 2 3 3 0 6 3 5 2 3 0 3 2 2 2 0 3 3 1 0 4 2 3 3 3 3 4 3 0 3 2 4 0 4
 2 1 1 0 3 0 2 4 1 1 4 1 0 0 6 0 0 3 4 1 0 1 1 2 4 2]
```

+ 코드 + Markdown

기존 Code 분석 및 토의

CNN (Convolutional Neural Networks) 소개

CNN의 개념

CNN은 주로 이미지나 영상 데이터를 처리할 때 사용되는 딥러닝 모델로, 핵심 연산인 '합성곱(Convolution)'을 통해 이미지의 중요한 특성을 추출한다.

이러한 과정을 통해 이미지 내 공간적 정보를 보존하면서도 유용한 특징만을 강조할 수 있게 된다.

CNN의 탄생 배경

전통적인 DNN(Deep Neural Network)은 1차원 데이터 처리에 최적화되어 있어, 2차원 이미지 데이터를 처리할 때 중요한 공간적 정보를 잃게 된다.

CNN은 이 문제를 해결하기 위해 개발되었으며, 이미지를 그대로 사용하여 공간적 정보의 손실 없이 특성을 추출한다.

CNN의 주요 특징

1. Locality (**지역성**)
   - CNN은 이미지 전체가 아닌 작은 부분을 보면서 인접한 픽셀 간의 상관관계를 비선형 필터를 통해 추출한다.
2. Shared Parameters (**매개변수 공유**)
   - 동일한 필터(커널)를 이미지 전체에 적용함으로써 매개변수의 수를 대폭 줄이고, 이미지의 다양한 위치에서 유사한 패턴을 효과적으로 인식할 수 있다.

Convolution의 작동 원리

기본 Convolution 과정

1. **필터 설정**
   - 입력 이미지에 적용할 필터(커널)의 크기와 특성을 정의.
2. **슬라이딩 윈도우**
   - 필터를 이미지 전체에 순차적으로 적용하면서, 각 위치에서 필터와 해당 부분의 내적을 계산.
3. **특성 맵 생성**
   - 필터 적용 결과를 모아 특징 맵(Feature Map)을 생성합니다. 이 맵은 원본 이미지의 특징을 강조한 결과 생성.

Zero Padding

- Convolution 과정에서 입력 이미지의 크기가 축소되는 것을 방지하기 위해 이미지의 주변을 0으로 채운다. 이를 통해 이미지의 크기를 유지할 수 있다.

Stride

- 필터를 적용하는 간격을 의미.
- Stride가 크면 특성 맵의 크기는 작아지며, 연산량도 줄어든다.

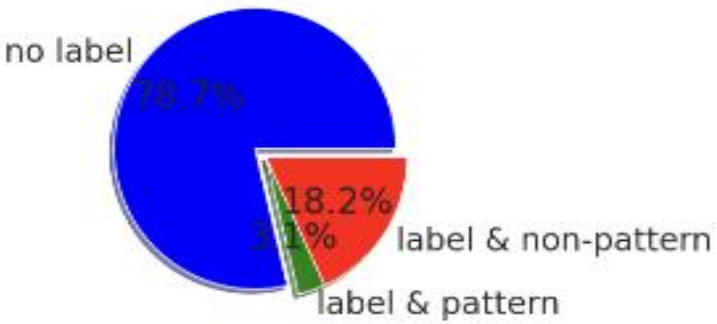
기존 Code 분석 및 토의

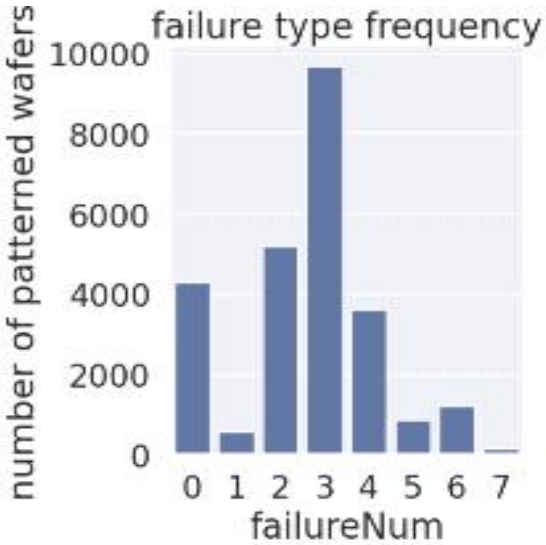
```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811457 entries, 0 to 811456
Data columns (total 5 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   waferMap        811457 non-null  object
 1   dieSize         811457 non-null  float64
 2   lotName         811457 non-null  object
 3   trianTestLabel  811457 non-null  object
 4   failureType     811457 non-null  object
dtypes: float64(1), object(4)
memory usage: 31.0+ MB
```

총 811,457개의 데이터셋
label x -> 638,507개
label o -> 172,950개

(label o)
pattern o -> 25,519개
pattern x -> 147,431개





label o & pattern o -> 3.1%에 불과

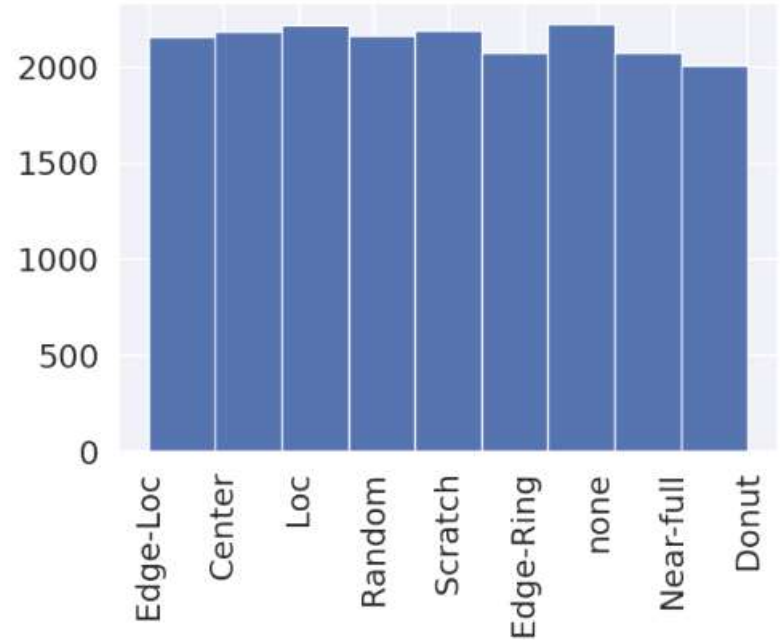
failure type frequency 불균형
-> overfitting 우려

기존 Code 분석 및 토의

```
[ ]  df_withlabel['waferMapDim'].value_counts() #use two dim (25,27) & (26,26)

waferMapDim
(25, 27)     18781
(26, 26)     14366
(30, 34)     12400
(29, 26)     11751
(27, 25)     10682
             ...
(68, 72)         1
(29, 45)         1
(38, 62)         1
(43, 133)        1
(36, 41)         1
Name: count, Length: 346, dtype: int64
```

label이 있는 데이터 中
waferMapDim이 (26, 26)인 것만
모델 학습 및 검증에 사용



노이즈를 추가하여
failure type 불균형 해결

기존 Code 분석 및 토의

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.keep_prob1 = 0.2
        self.keep_prob2 = 0.5

        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1), # 첫번째 conv layer : 입력 층 3, 출력 32, Relu, Pooling으로 MAX 적용
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))


        self.layer2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1), #2번째 conv layer : 입력 층 32, 출력 64, Rel., Pooling으로 MAX 적용
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))


        self.layer3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1), #3번째 conv layer : 입력 층 64, 출력 128, Relu, Pooling으로 Max 적용
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2, padding=1))

        self.fc1 = nn.Linear(3 * 3 * 128, 1253, bias=True) # fully connected,
        nn.init.xavier_uniform_(self.fc1.weight)
        self.layer4 = nn.Sequential(
            self.fc1,
            nn.ReLU()) #dropout 적용

        self.fc2 = nn.Linear(1253, 8, bias=True) #오픈씨큐 3개로 출력 8
        nn.init.xavier_uniform_(self.fc2.weight)


    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = out.view(out.size(0), -1)  # fully connect들어가서 flatten을 함.
        out = self.layer4(out)
        out = self.fc2(out)
        return out
```

데이터 전처리

↓

AutoEncoder

↓

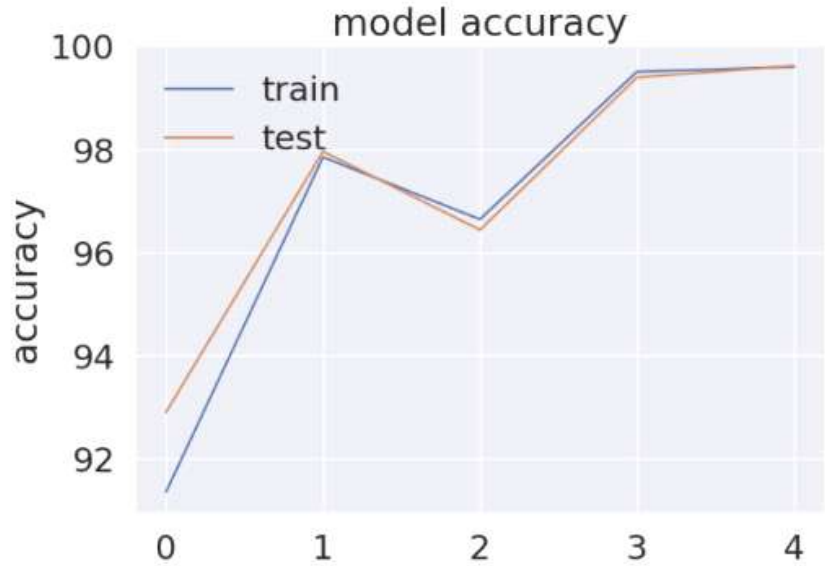
데이터 증강

↓

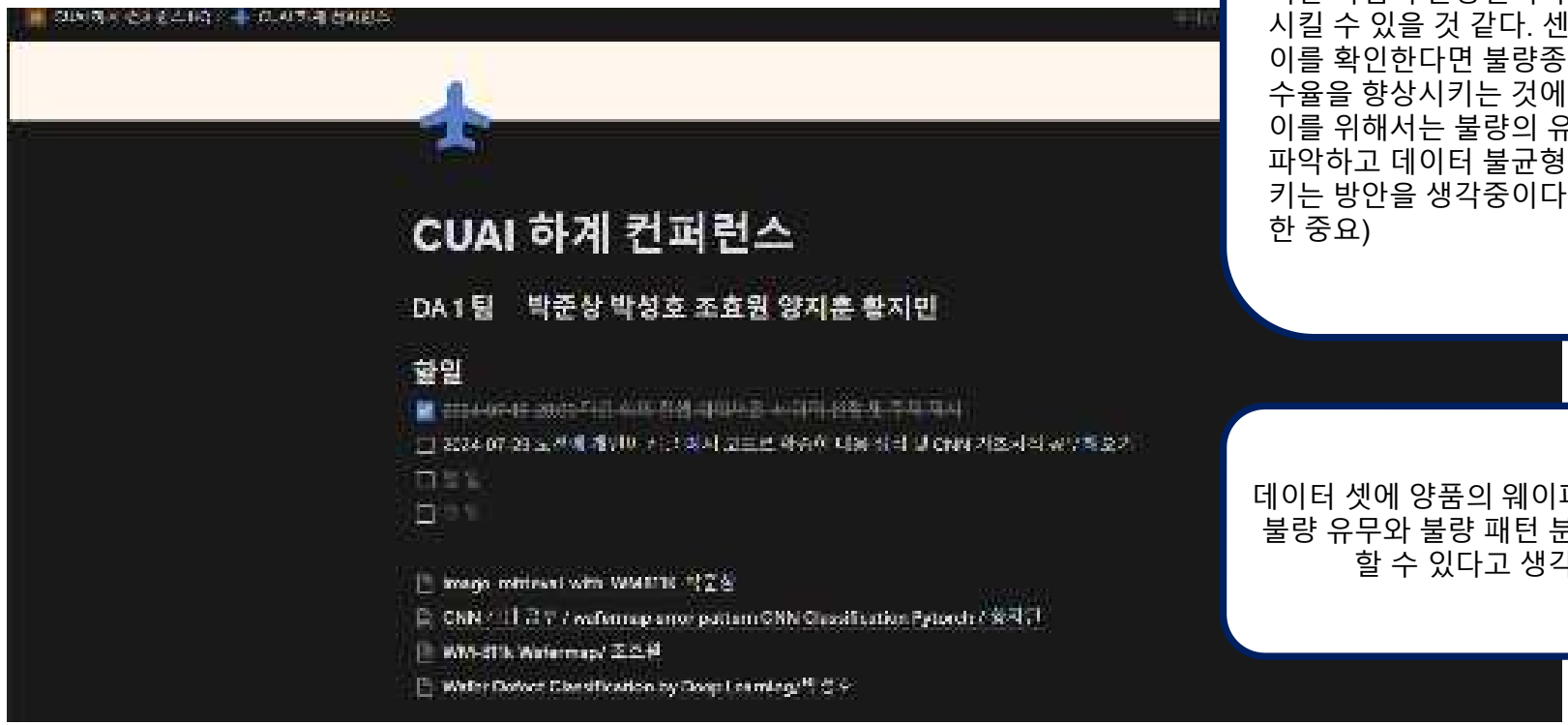
CNN

↓

train_test_split 및 K-Fold 검증



기존 Code 분석 및 토의



wafer의 불량 유/무를 예측하고 불량이 발생했다면 어떠한 타입의 불량인지까지를 예측하는 모델로 발전시킬 수 있을 것 같다. 센서 데이터를 통해 실시간으로 이를 확인한다면 불량종류에 따른 즉각적인 조치로 수율을 향상시키는 것에 기여할 수 있을 것이다.
이를 위해서는 불량의 유무에 따른 데이터의 개수를 파악하고 데이터 불균형을 고려하여 모델을 학습 시키는 방안을 생각중이다. (데이터를 처리하는 것 또한 중요)

데이터 셋에 양품의 웨이퍼를 추가한다면, 불량 유무와 불량 패턴 분석을 함께 수행할 수 있다고 생각합니다.

GAN과 DCGAN, sGAN, LSGAN을 이용해 모델을 비교한다.

Radon 변환을 CNN의 input으로 활용하는 방안의 가능성을 확인한다.

향후 계획

일자	계획
08/02	**DCGAN 및 sGAN 이론공부 및 논문분석 공유 / (26, 26)을 제외한 나머지 chip size 데이터 전처리 방향성 토의**
미정 **(-8/16**까지 마무리 계획**)**	**NIPA 서버를 활용해 CNN, GAN 알고리즘 모델 학습 및 개선 / 발표 영상 제작 및 논문 작성**
08/20	**예선 제출 마감**
08/27	**최종 발표**